

Oracle® AI Database

Oracle Data Provider for .NET Developer's Guide



26ai
G43543-03
November 2025

ORACLE®

Copyright © 2002, 2025, Oracle and/or its affiliates.

Primary Author: Maitreyee Chaliha

Contributing Authors: Alex Keh, Janis Greenberg, Kiminari Akiyama, Minal Agashe, Sumit Jeloka, Sinclair Hsu, Shailendra Jain, Riaz Ahmed, Ashish Shah, Lakshminarayanan Suriamoorthy, Steven Caminez, Naveen Doraiswamy, Neeraj Gupta, Chithra Ramamurthy, Martha Woo, Arun Singh, Sujith Somanathan, Nishant Singh

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface

Audience	i
Documentation Accessibility	i
Related Documents	i
Passwords in Code Examples	ii
Conventions	ii

Changes in This Release for Oracle Data Provider for .NET

Changes in Oracle Data Provider for .NET Release 23.26.0	ii
Changes in Oracle Data Provider for .NET Release 23.9	iii
Changes in Oracle Data Provider for .NET Release 23.8	vi
Changes in Oracle Data Provider for .NET Release 23.7	viii
Changes in Oracle Data Provider for .NET Release 23.6.1	xi
Changes in Oracle Data Provider for .NET Release 23.6	xi
Changes in Oracle Data Provider for .NET Release 23.5.1	xiv
Changes in Oracle Data Provider for .NET Release 23.5	xiv
Changes in Oracle Data Provider for .NET Release 23.3	xvi
Changes in Oracle Data Provider for .NET Release 21c (21.12)	xxiv
Changes in Oracle Data Provider for .NET Release 21c (21.8)	xxiv
Changes in Oracle Data Provider for .NET Release 21c (21.7)	xxv
Changes in Oracle Data Provider for .NET Release 21c (21.6.1)	xxvi
Changes in Oracle Data Provider for .NET Release 21c (21.5)	xxvi
Changes in Oracle Data Provider for .NET Release 21c (21.4.1)	xxvii
Changes in Oracle Data Provider for .NET Release 21c (21.4)	xxvii
Changes in Oracle Data Provider for .NET Release 21c (21.3)	xxviii
Changes in Oracle Data Provider for .NET (21.1)	xxix
Changes in Oracle Data Provider for .NET (19.16)	xxxi
Changes in Oracle Data Provider for .NET (19.15.1)	xxxi
Changes in Oracle Data Provider for .NET (19.10)	xxxii
Changes in Oracle Data Provider for .NET (19.9)	xxxiii
Changes in Oracle Data Provider for .NET ODAC Release 19c (19.3.2)	xxxiv
Changes in Oracle Data Provider for .NET (19.7)	xxxiv
Changes in Oracle Data Provider for .NET (19.6)	xxxv

Changes in Oracle Data Provider for .NET (19.5)	xxxv
Changes in Oracle Data Provider for .NET in ODAC Release 19c (19.3)	xxxvi
Changes in Oracle Data Provider for .NET Release 19c (19.3)	xxxvii

1 Introducing Oracle Data Provider for .NET

.NET Data Access in Oracle: Products and Documentation	1
Oracle Data Provider for .NET (ODP.NET)	1
Oracle Developer Tools for Visual Studio	2
Oracle Providers for ASP.NET	2
Oracle Services for Microsoft Transaction Server	2
Oracle TimesTen In-Memory Database	3
Overview of Oracle Data Provider for .NET (ODP.NET)	3
Oracle Data Provider for .NET Assemblies	4
Oracle Data Provider for .NET, Unmanaged Driver Assemblies	4
Oracle Data Provider for .NET, Managed Driver and ODP.NET Core Assemblies	4
Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces	5
Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client	5
Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Enumerations	12
Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces	15
Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Structures	15
Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Exceptions	15
Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Classes	16
Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Interfaces	17
Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Enumerations	18
Differences between the ODP.NET Drivers	18
Getting Started With Developing ODP.NET Applications	22

2 Installing and Configuring Oracle Data Provider for .NET

System Requirements	1
Entity Framework Requirements	3
Entity Framework Database First and Model First Requirements	3
Entity Framework Code First Requirements	3
Entity Framework Core System Requirements	3
Oracle Data Provider for .NET Versioning Scheme	4
Installing Oracle Data Provider for .NET, Unmanaged Driver	6
File Locations After Installation	6
Search Order for Unmanaged DLLs	7
Unmanaged ODP.NET and Dependent Unmanaged DLL Mismatch	7
Installing Oracle Data Provider for .NET, Managed Driver	7
Platform-Dependent Assemblies and Their Search Order	8

File Locations After Installation	9
Installing Oracle Data Provider for .NET Core	9
Entity Framework Assemblies and File Location	10
Configuring Oracle Data Provider for .NET	11
Oracle Client Configuration File Automated Setup During Installation	12
Oracle Client Configuration File Settings	12
Configuring .NET Framework to Use ODP.NET	14
ODP.NET Intellisense for .NET Configuration Files	17
Oracle Data Provider for .NET, Unmanaged Driver Configuration	17
Supported Configuration Settings	18
Windows Registry	25
Configuration File Support	25
SQL Translation Framework Configuration	27
Specifying UDT Mappings with Unified Configuration for Unmanaged ODP.NET	30
Oracle Data Provider for .NET, Managed Driver Configuration	31
version Section	32
dataSources Section	32
settings section	35
LDAPsettings section	41
Lightweight Directory Access Protocol	41
implicitRefCursor section	42
distributedTransaction section	42
connectionPools section	43
edmMappings section	43
onsConfig section	43
ONS TCPS and Wallets	44
Client Side ONS Daemon Configuration	45
Relative Windows Path and Windows Environment Variable Configuration Settings	46
Oracle Data Provider for .NET Core Configuration	47
Configuration Differences among ODP.NET Drivers	50
Centralized Configuration Providers for Deployments	51
Connection String URLs	52
Azure Centralized Configuration Provider	52
OCI Centralized Configuration Provider	54
Local File Centralized Configuration Provider	56
Vault Providers	57
Provider Configuration	60
Connection URL Caching	66
Connection Configuration Restriction	71
Configuring for Entity Framework Code First	75
Entity Framework 6 Code-Based Registration	76

Migrating from ODP.NET, Unmanaged Driver to ODP.NET, Managed Driver or ODP.NET Core	77
Configuring a Port to Listen for Database Notifications	77
General .NET Programming Recommendations and Tips for ODP.NET	78

3 Features of Oracle Data Provider for .NET

Base Classes, Provider Factory Classes, and DbDataSource	2
Code Access Security	6
Configuring OraclePermission	7
Configuring OraclePermission for Web Applications with High or Medium Trust Levels	8
Configuring OraclePermission for Windows Applications Running in a Partial Trust Environment	8
Connecting to Oracle Database	8
Connecting to Oracle Autonomous Database	9
Using Oracle Identity and Access Management	11
Configuring a Client Connection Using an IAM Database Password Verifier	11
Configuring a Client Connection Using Token	12
Configuring Client Connection to use Token File	12
Use of ODP.NET API to pass the Token	13
IAM Token Functionality Requirements	15
Sample Code	15
Configuring a Client Connection Using Database Alternate Password for Token Authentication	16
Comparing IAM Token Authentication Methods	16
IAM Single Sign-On	18
Using Microsoft Entra ID	21
Connection String Attributes and URLs	27
Connection String Builder	30
Specifying the Data Source Attribute	30
Using the TNS Alias	31
Using the Connect Descriptor	31
Easy Connect and Easy Connect Plus Naming Methods	31
Using LDAP	32
LDAP URL Naming	33
LDAPS Configuration without Wallets	33
Data Source Enumerator	34
Using WebSocket	34
Using Transport Layer Security and Secure Sockets Layer	35
Secure Sockets Layer and Transport Layer Security Differences	35
ODP.NET Secure Sockets Layer Configuration Using Wallets	36
ODP.NET Secure Sockets Layer Configuration without Wallets	38
Inserting Public Keys into System Trusted Certificate Authority List	38

Troubleshooting TLS/SSL Setup	39
Mutual TLS/SSL Certificate Selection	41
Using Secure External Password Store	42
Configuring Secure External Password Store (SEPS)	43
Using Kerberos	44
File Based Credential Cache and MSLSA	44
Using ODP.NET, Managed Driver with Kerberos	45
Configuring Kerberos Authentication with ODP.NET	45
Using Windows Native Authentication (NTS)	48
Configuring Windows Native Authentication (NTS) for the ODP.NET Client	49
Operating System Authentication Credentials	50
Network Data Encryption and Integrity	51
Using Data Encryption	51
Using Data Integrity	51
Schema Discovery	52
User Customization of Metadata	53
Connection Pooling	53
Using Connection Pooling	54
Connection Pool Management	55
Connection Performance Counters	56
Registering Performance Counters	56
Enabling Performance Counters	57
Setting Performance Counters in .NET Configuration Files	58
Setting Performance Counters in Windows Registry	59
Publishing Performance Counters	59
Database Resident Connection Pooling and Connection Manager in Traffic Director Mode	60
Oracle Multitenant and Pluggable Databases	63
Edition-Based Redefinition	64
Privileged Connections	66
Connection Pooling with OracleCredential	67
Securing Passwords with OracleOpaqueString	68
Password Expiration	69
Proxy Authentication	71
Dynamic Distributed Transaction Enlistment	72
Client Identifier and End-to-End Tracing	72
Automatic Connection Recovery	73
Reset Session State to Limit Application State Leaks	73
Real Application Clusters and Global Data Services	74
Fast Application Notification	75
In-Band Notification	76
Runtime Connection Load Balancing	76

Fast Connection Failover (FCF)	77
Using FCF Planned Outage to Minimize Service Disruption	78
Pool Behavior in an Oracle RAC Database	80
Using Transaction Guard to Prevent Logical Corruption	80
ODP.NET and Transaction Guard	81
Application Continuity	84
ODP.NET and Application Continuity	85
Transparent Application Failover	86
TAF Notification	88
When Failover Occurs	88
Registering an Event Handler for Failover	89
Oracle Globally Distributed Database	90
ODP.NET Sharding	91
OracleCommand Object	93
Command Auto-Commit	94
Transactions	94
System.Transactions and Promotable Transactions	95
Implicit Transaction Enlistment Using TransactionScope	95
Explicit Transaction Enlistment Using CommittableTransaction	96
Distributed Transactions	98
Microsoft Distributed Transaction Coordinator Integration	98
ODP.NET Core and Managed Driver Setup	99
ODP.NET, Unmanaged Driver Setup	99
Sessionless Transactions	100
Parameter Binding	102
Command Timeouts	102
OracleDbType Enumeration Type	103
Inference of DbType, OracleDbType, and .NET Types	104
PL/SQL Associative Array Binding	108
Array Binding	111
Batch Processing	114
Statement Caching	114
Statement Caching Connection String Attributes	115
Enabling Statement Caching through the Registry	115
Statement Caching Methods and Properties	115
Connections and Statement Caching	116
Pooling and Statement Caching	116
Self-Tuning	116
Self-Tuning Statement Caching	116
Enabling or Disabling Self-Tuning for Applications	118
Tracing Optimization Changes	118
Data Transmission Performance	119

Securing Identifiers and Literals from SQL Injection Attacks	119
ODP.NET Types Overview	120
GUIDs	122
Obtaining Data from an OracleDataReader Object	126
Typed OracleDataReader Accessors	126
.NET Type Accessors	126
ODP.NET Type Accessors	130
Obtaining LONG and LONG RAW Data	131
Setting InitialLONGFetchSize to Zero or a Value Greater than Zero	132
Setting InitialLONGFetchSize to -1	132
Obtaining LOB Data	132
Setting InitialLOBFetchSize to Zero	133
Setting InitialLOBFetchSize to a Value Greater than Zero	133
Setting InitialLOBFetchSize to -1	134
Performance Considerations Related to the InitialLOBFetchSize Property	135
Controlling the Number of Rows Fetched in One Database Round-Trip	136
Use of RowsToFetchPerRoundTrip	136
Use of FetchSize	136
Fine-Tuning FetchSize	137
Using the RowSize Property	137
PL/SQL REF CURSOR and OracleRefCursor	138
Obtaining an OracleRefCursor Object	138
Obtaining a REF CURSOR Data Type	139
Populating an OracleDataReader from a REF CURSOR	139
Populating the DataSet from a REF CURSOR	139
Populating an OracleRefCursor from a REF CURSOR	140
Updating a DataSet Obtained from a REF CURSOR	140
Behavior of ExecuteScalar Method for REF CURSOR	140
Passing a REF CURSOR to a Stored Procedure	141
Implicit REF CURSOR Binding	142
Specifying REF CURSOR Bind and Metadata Information in the .NET Configuration File	143
Sample Configuration File and Application	146
Usage Considerations	149
CommandText Property Considerations	149
Bind Considerations	149
Overloaded Stored Procedures	150
Type Initialization Exceptions	150
Using Stored Functions with Function Import	150
LOB Support	150
Large Character and Large Binary Data Types	151
Oracle Data Provider for .NET LOB Objects	151
Updating LOBs Using a DataSet	152

Updating LOBs Using OracleCommand and OracleParameter	152
Updating LOBs Using ODP.NET LOB Objects	153
Temporary LOBs	153
Native JSON Support	154
Artificial Intelligence Vectors and Semantic Search	156
ODP.NET XML Support	165
Supported XML Features	166
XQuery Support	166
OracleXmlType and Connection Dependency	167
Updating XMLType Data in the Database	167
Updating with DataSet, OracleDataAdapter, and OracleCommandBuilder	168
Updating with OracleCommand and OracleParameter	168
Updating XML Data in OracleXmlType	169
Characters with Special Meaning in XML	169
Retrieving Query Result Set as XML	169
Handling Date and Time Format	170
Characters with Special Meaning in Column Data	170
Characters in Table or View Name	171
Case-Sensitivity in Column Name to XML Element Name Mapping	171
Column Name to XML Element Name Mapping	172
Object-Relational Data	173
NULL Values	173
Data Manipulation Using XML	173
Handling Date and Time Format	174
Saving Changes Using XML	174
Characters with Special Meaning in Column Data	174
Characters with Special Meaning in Table or View Name	175
Case-Sensitivity in XML Element Name to Column Name Mapping	175
XML Element Name to Column Name Mapping	175
Saving Changes to a Table Using an XML Document	176
Object-Relational Data	177
Multiple Tables	177
Commit Transactions	177
Oracle User-Defined Types (UDTs) and .NET Custom Types	178
Oracle User-Defined Types (UDTs)	178
Custom Types	179
Required Custom Type Implementations	179
Optional Custom Type Implementations	180
Specifying Custom Type Mappings	181
Using a Custom Type Factory to Specify Custom Type Mappings	182
Using XML in Configuration Files to Specify Custom Type Mappings	182
Using Custom Type Mappings	183

Converting Between Custom Types and Oracle UDTs	184
Oracle UDT Attribute Mappings	185
Oracle UDT Retrieval from OracleDataReader	186
Oracle UDT Metadata Retrieval from OracleDataReader	187
Oracle UDT Parameter Binding with OracleParameter	188
Guidelines for Binding UDT Input and Output Parameters	188
UDT Input Parameter Binding with OracleParameters	189
UDT Output Parameter Binding with OracleParameters	190
Populating the DataSet with Oracle UDTs	191
UDT Method Invocation	192
Configuration Settings for Oracle UDTs	192
StatementCacheWithUdts	193
UdtCacheSize	193
Using UDTs with Managed ODP.NET and ODP.NET Core	193
Migrating from Unmanaged ODP.NET to Managed or Core	195
Handling NULL Attribute Values in UDTs	196
Bulk Copy	196
Data Types Supported by Bulk Copy	197
Restrictions on Oracle Bulk Copy of a Single Partition	198
Integrity Constraints Affecting Oracle Bulk Copy	198
Database Insert Triggers	198
Field Defaults	198
Asynchronous Programming and Pipelining	199
Advanced Queuing and Transactional Event Queues	204
Using ODP.NET for Advanced Queuing	206
Enqueuing and Dequeuing Example	207
Continuous Query Notification Support	209
Client Initiated Continuous Query Notifications	211
Continuous Query Notification Classes	211
Supported Operations	212
Requirements of Notification Registration	213
Using Continuous Query Notification	213
Application Steps	213
Flow of Notification Process	214
Best Practice Guidelines and Performance Considerations	216
OracleDataAdapter Safe Type Mapping	216
Comparison Between Oracle Data Types and .NET Types	217
SafeMapping Property	218
Using Safe Type Mapping	218
OracleDataAdapter Requery Property	220
Guaranteeing Uniqueness in Updating DataSet to Database	221
What Constitutes Uniqueness in DataRow Objects?	221

Configuring PrimaryKey and Constraints Properties	222
Updating Without PrimaryKey and Constraints Configuration	222
Globalization Support	223
Globalization Settings	223
Client Globalization Settings	223
Session Globalization Settings	224
Thread-Based Globalization Settings	224
Globalization-Sensitive Operations	226
Operations Dependent on Client Computer's Globalization Settings	226
Operations Dependent on Thread Globalization Settings	226
Operations Sensitive to Session Globalization Parameters	226
ODP.NET Driver Globalization Differences	227
Data Use Case Domains	227
Annotations	227
OpenTelemetry	228
.NET Metrics	235
Debug Tracing	237
Database Application Migration: SQL Translation Framework	239
The SQL Translation Profile	239
Allowed Logon Version Client Support	241

4 ADO.NET Entity Framework and LINQ to Entities

Overview of Entity Framework	1
Language Integrated Query and Entity SQL	2
Mapping Oracle Data Types to EDM Types	2
EDM Type Facets	6
Oracle Number Default Data Type Mapping and Customization	13
Entity Framework 6 Mapping and Customization	14
New Default Mappings	14
Data Type Mapping and Customization Process	15
StoreGeneratedPattern Enumeration	16
Identity Attribute	16
Virtual Column	16
Resolving Compilation Errors When Using Custom Mapping	16
Mapping Boolean and Guid Parameters in Custom INSERT, UPDATE, and DELETE	16
Stored Procedures	16
Migrating Existing Entity Framework 5 Applications to Entity Framework 6	17
Code First	17
Mapping of .NET Types to Oracle Types	18
Code First Migrations	21
Code First Migrations With No Supporting Code Migration File	22

Code First Database Initialization	22
Oracle Database Object Creation	23
Using the Default Connection Factory	25
Unsupported Entity Framework Features	25

5 Oracle Data Provider for .NET Entity Framework Core

Oracle Entity Framework Core 10 Features	1
Oracle Entity Framework Core 9 Features	3
Oracle Entity Framework Core 8 Features	5
Oracle Entity Framework Core 7 Features	7
Application Programming Interface	9
DatabaseFacade Class	9
DatabaseFacade.IsOracle	9
DatabaseFacade.EnsureCreated	10
DatabaseFacade.EnsureCreated(string[])	10
DatabaseFacade.EnsureDeleted	11
DatabaseFacade.EnsureDeleted(string[])	12
DbContextOptionsBuilder Class	12
OracleDbContextOptionsBuilder.UseNetTopologySuite(double tolerance)	13
DbContextOptionsBuilder.UseOracle	13
UseOracle(string connectionString)	13
UseOracle(DbContextOptionsBuilder, Action<OracleDbContextOptionsBuilder> oracleOptionsAction = null)	14
UseOracleSQLCompatibility(enum version)	14
IQueryingEnumerable Interface	15
IQueryingEnumerable.ToQueryString Extension Method	15
MigrationBuilder Class	16
ModelBuilder Class	16
ModelBuilder UseIdentityColumn() and UseOracleIdentityColumn()	16
OracleSQLCompatibility Enumeration	17
Database Functions (EF.Functions) for Vectors	17
VectorDistance	17
VectorEmbedding	19
ToVector	20
Sample Code	21
Using ODP.NET Core Classes	22
Logging	22
Migrations	24
Scaffolding Or Reverse Engineering	26
Identifier Name Length and Uniqueness	31
Using Large Character or Binary Data Types	32

Performance Considerations	32
Artificial Intelligence Vector Data Types	33
Spatial Data and NetTopologySuite	39
Breaking Changes	45

6 Oracle Data Provider for .NET Classes

OracleAccessToken Class	2
OracleAccessToken Members	3
OracleAccessToken Constructors	4
OracleAccessToken(char)	4
OracleAccessToken(char, char)	5
RefreshAccessToken Event	5
OracleClientFactory Class	6
OracleClientFactory Members	7
OracleClientFactory Field	9
Instance	9
OracleClientFactory Constructor	10
OracleClientFactory Public Properties	10
CanCreateDataSourceEnumerator	10
OracleClientFactory Public Methods	11
CreateCommand	12
CreateCommandBuilder	12
CreateConnection	13
CreateConnectionStringBuilder	13
CreateDataAdapter	13
CreateDataSource	14
CreateDataSourceEnumerator	14
CreateParameter	15
CreatePermission	15
OracleCommand Class	16
OracleCommand Members	18
OracleCommand Constructors	21
OracleCommand()	22
OracleCommand(string)	22
OracleCommand(string, OracleConnection)	23
OracleCommand Static Methods	23
OracleCommand Properties	23
AddRowid	25
AddToStatementCache	26
ArrayBindCount	27
ArrayBindRowsAffected	28

BindByName	29
CommandText	30
CommandTimeout	31
CommandType	32
Connection	32
DesignTimeVisible	33
FetchSize	33
FetchSqlId	34
ImplicitRefCursors	35
InitialLOBFetchSize	36
InitialLONGFetchSize	36
Notification	37
NotificationAutoEnlist	38
Parameters	39
RowSize	40
RowsToFetchPerRoundTrip	40
SqlId	41
Transaction	42
UpdatedRowSource	42
UseEdmMapping	43
XmlCommandType	43
XmlQueryProperties	44
XmlSaveProperties	45
OracleCommand Public Methods	46
Cancel	47
Clone	51
CreateParameter	51
Dispose	52
ExecuteNonQuery	52
ExecuteNonQueryAsync	54
ExecuteNonQueryAsync()	55
ExecuteNonQueryAsync(CancellationTokentoken)	55
ExecuteReader	56
ExecuteReader()	57
ExecuteReader(CommandBehavior)	58
ExecuteReaderAsync	59
ExecuteReaderAsync()	61
ExecuteReaderAsync(CancellationTokentoken)	61
ExecuteReaderAsync(CommandBehavior behavior)	62
ExecuteReaderAsync(CommandBehavior behavior, CancellationTokentoken)	63
ExecuteScalar	63

ExecuteStream	65
ExecuteToStream	66
ExecuteScalarAsync	66
ExecuteScalarAsync()	67
ExecuteScalarAsync(CancellationToken cancellationToken)	68
ExecuteXmlReader	69
ExecuteXmlReaderAsync	70
ExecuteXmlReaderAsync()	71
ExecuteXmlReaderAsync(CancellationToken)	71
Prepare	72
OracleCommandBuilder Class	72
OracleCommandBuilder Members	74
OracleCommandBuilder Constructors	77
OracleCommandBuilder()	77
OracleCommandBuilder(OracleDataAdapter)	77
OracleCommandBuilder Static Methods	78
DeriveParameters	78
OracleCommandBuilder Properties	81
CaseSensitive	82
CatalogLocation	82
CatalogSeparator	83
ConflictOption	83
DataAdapter	84
QuotePrefix	84
QuoteSuffix	85
SchemaSeparator	85
OracleCommandBuilder Public Methods	86
GetDeleteCommand	87
GetDeleteCommand()	88
GetDeleteCommand(bool)	88
GetInsertCommand	89
GetInsertCommand()	90
GetInsertCommand(bool)	90
GetUpdateCommand	91
GetUpdateCommand()	92
GetUpdateCommand(bool)	92
QuoteIdentifier	93
RefreshSchema	94
UnquoteIdentifier	94
OracleCommandBuilder Events	95
OracleConfiguration Class	96
OracleConfiguration Members	98

OracleConfiguration Methods	105
DisableConnectionStringAllowedProperties	105
EnableConnectionStringAllowedProperties	106
EnableConnectionStringAllowedProperties(string, bool)	106
EnableConnectionStringAllowedProperties(FileInfo, bool)	107
OracleConfiguration Connection Properties	108
ConfigurationProviders	110
DatabaseCharset	111
DatabaseNCharset	112
DatabaseEditionName	113
DisableOOB	113
DrpcConnectionClass	113
HostnameDefaultServiceIsHost	114
MapVectorColumnAsClob	114
NamesDirectoryPath	115
OciCompartment	115
OciConfigurationFile	116
OciDatabase	117
OciUrl	117
OciProfile	118
OciTenancy	118
OracleDataSources	119
SuppressGetDecimalInvalidCastException	119
TAFUsePooledConnections	120
TcpConnectTimeout	121
TimeZoneFileLocation	121
TnsAdmin	122
UseSNI	123
OracleConfiguration Secure Connection Properties	124
AllowUncPaths	125
PasswordAuthentication	126
SEPSWalletLocation	126
SEPSWalletPassword	127
SqlNetAllowedLogonVersionClient	127
SqlNetAllowWeakCrypto	128
SqlNetAuthenticationServices	129
SqlNetCloudUser	129
SqlNetCryptoChecksumClient	130
SqlNetCryptoChecksumTypesClient	131
SqlNetEncryptionClient	131
SqlNetEncryptionTypesClient	132
SqlNetIgnoreANOEncryptionForTCPS	132

SqlNetWalletOverride	133
SqlNetURI	134
SSLCertificateThumbprint	134
SSLServerDNMatch	135
SSLVersion	135
TLSExcludeVersion	136
TLSVersion	137
TokenAuthentication	138
TokenLocation	139
UseMITKerberos	140
WalletLocation	140
WalletPassword	141
OracleConfiguration Command Properties	142
AddOracleTypesDeserialization	143
BindByName	144
CommandTimeout	144
GetDecimalRetainTrailingZeros	145
ProviderTypeNLSSensitiveToString	146
UseClientInitiatedCQN	146
OracleConfiguration Directories Properties	147
DefaultAdminContext	148
DirectoryServers	148
DirectoryServerType	149
LdapAdmin	149
NamesLdapAuthenticateBind	150
NamesLdapAuthenticateBindMethod	151
NamesLdapConnTimeout	151
OracleConfiguration High Availability Properties	152
ChunkMigrationConnectionTimeout	153
DbNotificationAddress	154
DbNotificationPort	155
DistributedTransactionRecoveryAddress	155
DistributedTransactionRecoveryPort	156
DistributedTransactionTimeToLive	156
HAEvents	157
MaxCallsPerRequestBoundary	157
OnsConfigFile	158
OnsMode	158
OnsProtocol	159
OnsWalletLocation	160
OracleOnsServers	160
ServiceRelocationConnectionTimeout	161

OracleConfiguration Performance Properties	161
DefaultSDUSize	162
FetchSize	163
LoadBalancing	163
MaxStatementCacheSize	164
PerformanceCounters	164
Pipelining	165
ReceiveBufferSize	165
RowsToFetchPerRoundTrip	165
SelfTuning	166
SendBufferSize	167
StatementCacheSize	167
TcpNoDelay	167
OracleConfiguration Diagnostics and Tracing Properties	168
ConnectionIdPrefix	169
OpenTelemetryTracing	169
PoolNames	170
Program	170
SuppressErrorURL	171
TraceFileLocation	171
TraceFileMaxSize	172
TraceLevel	173
TraceOption	173
OracleConfigurationProvider Class	174
OracleConnection Class	175
OracleConnection Members	177
OracleConnection Constructors	185
OracleConnection()	186
OracleConnection(string)	187
OracleConnection(string, OracleCredential)	187
OracleConnection Static Methods	188
ClearAllPools	189
ClearPool	190
ClearInteractiveTokenCache	191
UnregisterCloudConfigNotification	192
OracleConnection Properties	193
AccessToken	198
ActionName	199
AllowCertificateSelectionUI	200
AutoCommit	201
AutoProxy	202
BindByName	203

ChunkMigrationConnectionTimeout	204
ClientId	205
ClientInfo	205
ClientTimeZoneFileVersion	206
CommandTimeout	207
ConnectionIdPrefix	208
ConnectionString	208
ConnectionTimeout	214
ConnectionType	215
Credential	215
Database	216
DatabaseCharset	217
DatabaseNCharset	217
DatabaseDomainName	218
DatabaseEditionName	218
DatabaseMaxIdentifierLength	219
DatabaseName	219
DatabaseTimeZoneFileVersion	220
DataSource	220
DRCPConnectionClass	221
DRCPPurity	222
FetchSize	223
HostName	224
HttpsProxy	224
HttpsProxyPort	225
InstanceName	225
IsInTransaction	226
IsNNEChecksummed	226
IsNNEEncrypted	227
IsTLSEncrypted	227
KeepAlive	228
KeepAliveInterval	228
KeepAliveTime	229
MaxStatementCacheSize	230
ModuleName	230
NamesDirectoryPath	231
OciCompartment	232
OciDatabase	232
OciIamUrl	233
OciTenacy	233
PasswordAuthentication	234
PDBName	234

Pipelining	235
PoolName	236
ProviderName	237
ProviderNuGetVersion	237
ProviderVersion	238
RemoteConfigurationFiltering	238
Roles	239
RowsToFetchPerRoundTrip	240
SDU	241
ServerVersion	241
ServiceName	242
ServiceRelocationConnectionTimeout	243
SessionlessTransactionId	244
SessionlessTransactionOptions	244
SqlNetAllowedLogonVersionClient	245
SqlNetAuthenticationServices	246
SqlNetCryptoChecksumClient	246
SqlNetCryptoChecksumTypesClient	247
SqlNetEncryptionClient	248
SqlNetEncryptionTypesClient	249
SqlNetWalletOverride	249
SSLCertificateThumbprint	250
SSLServerCertDN	251
SSLServerDNMatch	251
SSLVersion	252
State	252
StatementCacheSize	253
SuppressGetDecimalInvalidCastException	254
SwitchedConnection	255
TAFMode	255
TLSExcludeVersion	256
TLSVersion	257
TnsAdmin	257
TokenAuthentication	258
TokenLocation	260
UseClientInitiatedCQN	261
UseHourOffsetForUnsupportedTimezone	262
UseSNI	263
WalletLocation	264
WalletPassword	265
OracleConnection Public Methods	266
BeginRequest	268

BeginSessionlessTransaction	269
BeginTransaction	269
BeginTransaction()	270
BeginTransaction(IsolationLevel)	271
ChangeDatabase	272
Clone	273
Close	274
CreateCommand	275
Commit()	276
DisableReplay	276
EndRequest	277
EnlistDistributedTransaction	278
EnlistTransaction	280
FlushCache	281
GetSchema	281
GetSchema()	282
GetSchema (string collectionName)	283
GetSchema (string collectionName, string[] restrictions)	285
GetSessionInfo	287
GetSessionInfo()	288
GetSessionInfo(OracleGlobalization)	289
Open	289
OpenAsync	290
OpenAsync()	291
OpenAsync(Cancellation_token cancellationToken)	291
OpenWithNewClientSecret	293
OpenWithNewPassword	294
OpenWithNewPassword(String)	294
OpenWithNewPassword(SecureString)	295
OpenWithNewPassword(OracleOpaqueString)	296
OpenWithNewPasswordAsync	296
OpenWithNewPasswordAsync(String)	297
OpenWithNewPasswordAsync(string, Cancellation_token)	298
OpenWithNewPasswordAsync(SecureString)	299
OpenWithNewPasswordAsync(SecureString, Cancellation_token)	300
OpenWithNewPasswordAsync(OracleOpaqueString)	301
OpenWithNewPasswordAsync(OracleOpaqueString, Cancellation_token)	301
OpenWithNewToken	302
OpenWithNewToken(char[])	302
OpenWithNewToken(char[], char[])	303
PurgeStatementCache	304
ResumeSessionlessTransaction	305

Rollback	306
Rollback()	306
Rollback(String)	307
Save(String)	308
SetSessionInfo	309
SetShardingKey(OracleShardingKey, OracleShardingKey)	310
SuspendSessionlessTransaction	311
UseAzureTokenAuthentication	312
UseOciTokenAuthentication	312
OracleConnection Events	313
ConnectionOpen	314
Failover	315
HAEvent	316
InfoMessage	317
StateChange	318
OracleConnectionOpenEventArgs Class	318
OracleConnectionOpenEventArgs Members	319
OracleConnectionOpenEventArgs Properties	320
Connection	320
ConnectionOpenReason	321
OracleConnectionOpenEventHandler Delegate	321
OracleConnectionStringBuilder Class	322
OracleConnectionStringBuilder Members	325
OracleConnectionStringBuilder Constructors	328
OracleConnectionStringBuilder()	328
OracleConnectionStringBuilder(string)	328
OracleConnectionStringBuilder Public Properties	329
ConnectionLifeTime	331
ConnectionTimeout	332
ContextConnection	332
DataSource	333
DBAPrivilege	333
DecrPoolSize	334
Enlist	335
HAEvents	335
IncrPoolSize	336
IsFixedSize	336
Item	337
Keys	337
LoadBalancing	338
MaxPoolSize	338
MetadataPooling	339

MinPoolSize	339
Password	340
PersistSecurityInfo	340
Pooling	341
ProxyPassword	341
ProxyUserId	342
SelfTuning	342
StatementCachePurge	343
StatementCacheSize	343
TnsAdmin	344
TokenAuthentication	344
TokenLocation	345
UserID	346
ValidateConnection	346
Values	347
WalletLocation	347
WalletPassword	348
OracleConnectionStringBuilder Public Methods	348
Clear	349
ContainsKey	349
Remove	350
TryGetValue	351
OracleCredential Class	352
OracleCredential Members	353
OracleCredential Constructors	354
OracleCredential(string, SecureString)	355
OracleCredential(string, SecureString, OracleDBAPrivilege)	356
OracleCredential(string, SecureString, string, SecureString)	356
OracleCredential Properties	357
DBAPrivilege	357
Password	358
ProxyPassword	358
ProxyUserId	359
UserId	359
OracleCredential Public Methods	360
SetPassword	360
SetProxyPassword	361
OracleDataAdapter Class	362
OracleDataAdapter Members	364
OracleDataAdapter Constructors	366
OracleDataAdapter()	367
OracleDataAdapter(OracleCommand)	367

OracleDataAdapter(string, OracleConnection)	368
OracleDataAdapter(string, string)	368
OracleDataAdapter Static Methods	369
OracleDataAdapter Properties	369
DeleteCommand	371
IdentityInsert	371
IdentityUpdate	372
InsertCommand	373
Requery	373
ReturnProviderSpecificTypes	374
SafeMapping	374
SelectCommand	375
SuppressGetDecimalInvalidCastException	376
UpdateBatchSize	377
UpdateCommand	378
OracleDataAdapter Public Methods	378
Fill	379
Fill(DataTable, OracleRefCursor)	380
Fill(DataSet, OracleRefCursor)	381
Fill(DataSet, string, OracleRefCursor)	381
Fill(DataSet, int, int, string, OracleRefCursor)	382
OracleDataAdapter Events	383
RowUpdated	384
RowUpdating	386
OracleDatabase Class	387
OracleDatabase Members	389
OracleDatabase Constructors	390
OracleDatabase Constructor(string)	390
OracleDatabase Constructor(string, OracleCredential)	391
OracleDatabase Properties	391
ServerVersion	392
OracleDatabase Public Methods	392
Dispose	393
ExecuteNonQuery	393
Shutdown	394
Shutdown()	394
Shutdown(OracleDBShutdownMode, bool)	395
Startup	397
Startup()	398
Startup(OracleDBStartupMode, bool)	398
Startup(OracleDBStartupMode, string, bool)	399
OracleDataReader Class	400

OracleDataReader Members	403
OracleDataReader Static Methods	407
OracleDataReader Properties	408
Depth	409
FetchSize	409
FieldCount	410
HasRows	411
HiddenFieldCount	412
IsClosed	413
Item	413
Item [index]	414
Item [string]	415
InitialLOBFetchSize	415
InitialLONGFetchSize	416
RecordsAffected	417
RowSize	417
RowsToFetchPerRoundTrip	418
SuppressGetDecimalInvalidCastException	419
UseEdmMapping	420
VisibleFieldCount	420
OracleDataReader Public Methods	422
Close	425
Dispose	426
GetBoolean	426
GetByte	427
GetByteArray(int)	428
GetByteArray(string)	429
GetBytes	430
GetChar	431
GetChars	431
GetData	433
GetDataTypeName	433
GetDateTime	434
GetDateTimeOffset	435
GetDecimal	435
GetDouble	436
GetDoubleArray(int)	437
GetDoubleArray(string)	438
GetEnumerator	439
GetFieldType	439
GetFieldValueAsync	440
GetFieldValueAsync(int, CancellationToken)	440

GetFieldValueAsync(int)	441
GetFloat	442
GetFloatArray(int)	443
GetFloatArray(string)	443
GetGuid	444
GetInt16	445
GetInt16Array(int)	446
GetInt16Array(string)	447
GetInt32	448
GetInt64	448
GetName	449
GetNonZeroDoubleArray	450
GetNonZeroDoubleArray(int)	451
GetNonZeroDoubleArray(string)	451
GetNonZeroFloatArray	452
GetNonZeroFloatArray(int)	453
GetNonZeroFloatArray(string)	454
GetNonZeroIndexArray	454
GetNonZeroIndexArray(int)	455
GetNonZeroIndexArray(string)	456
GetNonZeroInt16Array	456
GetNonZeroInt16Array(int)	457
GetNonZeroInt16Array(string)	458
GetOracleBFile	458
GetOracleBinary	459
GetOracleBlob	460
GetOracleBlobForUpdate	461
GetOracleBlobForUpdate(int)	461
GetOracleBlobForUpdate(int, int)	463
GetOracleBoolean(int)	465
GetOracleClob	466
GetOracleClobForUpdate	466
GetOracleClobForUpdate(int)	467
GetOracleClobForUpdate(int, int)	469
GetOracleDataReader	471
GetOracleDate	472
GetOracleDecimal	473
GetOracleIntervalDS	473
GetOracleIntervalYM	474
GetOracleRef	475
GetOracleRefCursor	476
GetOracleString	477

GetOracleTimeStamp	478
GetOracleTimeStampLTZ	479
GetOracleTimeStampTZ	480
GetOracleXmlType	480
GetOracleValue	481
GetOracleValues	482
GetOracleVector	483
GetOracleVector(int)	483
GetOracleVector(string)	484
GetOrdinal	485
GetProviderSpecificFieldType	485
GetProviderSpecificValue	486
GetProviderSpecificValues	487
GetSchemaTable	488
GetString	493
GetTimeSpan	494
GetValue	495
GetValues	496
GetVectorDimensions	497
GetVectorDimensions(int)	497
GetVectorDimensions(string)	498
GetVectorNumericFormat(int)	498
GetVectorNumericFormat(string)	499
GetXmlReader	500
IsDBNull	501
IsDBNullAsync	502
IsDBNullAsync(int)	502
IsDBNullAsync(int, CancellationToken)	503
NextResult	504
NextResultAsync	505
NextResultAsync()	505
NextResultAsync(CancellationToken)	506
Read	507
ReadAsync	508
ReadAsync()	508
ReadAsync(CancellationToken)	509
OracleDataSource Class	510
OracleDataSource Members	511
OracleDataSource Public Methods	512
ClearPool	513
CreateCommand	513
CreateConnection	514

OpenConnection	515
OpenConnectionAsync	515
OracleDataSource Properties	516
ConnectionString	516
DatabaseCharset	517
DatabaseMaxIdentifierLength	517
DatabaseNCharset	518
Password	518
ProxyPassword	519
RetryCount	519
RetryDelay	520
OracleDataSource Public Events	520
OracleDataSourceBuilder Class	521
OracleDataSourceBuilder Members	522
OracleDataSourceBuilder Constructor	523
OracleDataSourceBuilder Public Methods	523
Build	524
UsePassword	524
UsePassword(password)	524
UsePassword(opaquePassword)	525
UseProxyPassword	526
UseProxyPassword(proxyPassword)	526
UseProxyPassword(opaquePassword)	527
OracleDataSourceBuilder Properties	527
ConnectionString	528
Methods, Properties, and Events in Common with OracleConnection or OracleDataSource	528
OracleDataSourceCollection Class	532
OracleDataSourceCollection Members	534
OracleDataSourceCollection Methods	535
Add	535
Remove	536
OracleDataSourceCollection Properties	536
Count	537
This	537
OracleDataSourceCommand Class	538
OracleDataSourceCommand Members	539
OracleDataSource Public Methods	539
ExecuteReader	540
ExecuteReaderAsync	541
ExecuteReaderAsync(cancellation_token)	541
ExecuteReaderAsync(commandBehavior, cancellation_token)	542

OracleDataSourceCommand Properties	542
OracleDataSourceEnumerator Class	543
OracleDataSourceEnumerator Members	544
OracleDataSourceEnumerator Constructor	545
OracleDataSourceEnumerator Public Methods	545
GetDataSources	546
OracleDBMSAssert Class	547
OracleDBMSAssert Methods	547
EnquoteIdentifier	548
EnquoteIdentifier(identifier, maxIdentifierLength, charset, bAlwaysQuote, bCapitalize)	548
EnquoteIdentifier(identifier, bAlwaysQuote, bCapitalize)	551
EnquoteLiteral	553
EnquoteLiteral(literal, charset)	553
EnquoteLiteral(literal)	555
EnquoteNCharLiteral	556
EnquoteNCharLiteral(literal, ncharset)	556
EnquoteNCharLiteral(literal)	557
OracleError Class	558
OracleError Members	560
OracleError Static Methods	561
OracleError Properties	561
ArrayBindIndex	561
DataSource	562
Message	563
Number	563
Procedure	564
Source	564
OracleError Methods	564
ToString	565
OracleErrorCollection Class	565
OracleErrorCollection Members	567
OracleErrorCollection Static Methods	568
OracleErrorCollection Properties	568
OracleErrorCollection Public Methods	569
OracleException Class	569
OracleException Members	571
OracleException Static Methods	572
OracleException Properties	573
DataSource	574
Errors	574
IsRecoverable	574

Message	575
Number	576
OracleLogicalTransaction	576
Procedure	577
Source	578
OracleException Methods	578
GetObjectData	579
ToString	579
OracleInfoMessageEventArgs Class	580
OracleInfoMessageEventArgs Members	582
OracleInfoMessageEventArgs Static Methods	583
OracleInfoMessageEventArgs Properties	583
Errors	584
Message	584
Source	585
OracleInfoMessageEventArgs Public Methods	585
ToString	585
OracleInfoMessageEventHandler Delegate	586
OracleLogicalTransaction Class	587
OracleLogicalTransaction Members	588
OracleLogicalTransaction Public Read-Only Properties	589
Committed	589
ConnectionString	590
LogicalTransactionId	591
UserCallCompleted	591
OracleLogicalTransaction Methods	592
Dispose	592
GetOutcome	593
OracleOnsServerCollection Class	594
OracleOnsServerCollection Members	595
OracleOnsServerCollection Methods	596
Add	596
Remove	597
OracleOnsServerCollection Properties	598
Count	598
This	598
OracleOpaqueString Class	599
OracleOpaqueString Members	600
OracleOpaqueString Constructor	601
OracleOpaqueString Properties	601
Length	601
OracleOpaqueString Methods	602

AppendChar	602
Clear	603
Copy	603
Dispose	604
IsReadOnly	604
MakeReadOnly	604
OracleParameter Class	605
OracleParameter Members	607
OracleParameter Constructors	609
OracleParameter()	610
OracleParameter(string, OracleDbType)	611
OracleParameter(string, object)	612
OracleParameter(string, OracleDbType, ParameterDirection)	613
OracleParameter(string, OracleDbType, object, ParameterDirection)	614
OracleParameter(string, OracleDbType, int)	615
OracleParameter(string, OracleDbType, int, string)	616
OracleParameter(string, OracleDbType, int, ParameterDirection, bool, byte, byte, string, DataRowVersion, object)	617
OracleParameter(string, OracleDbType, int, object, ParameterDirection)	619
OracleParameter Static Methods	620
OracleParameter Properties	621
ArrayBindSize	622
ArrayBindStatus	623
CollectionType	624
DbType	625
Direction	626
IsNullable	626
Offset	627
OracleDbType	628
OracleDbTypeEx	628
ParameterName	629
Precision	630
Scale	630
Size	631
SourceColumn	633
SourceColumnNullMapping	633
SourceVersion	634
Status	634
UdtTypeName	635
Value	636
OracleParameter Public Methods	638
Clone	638

Dispose	640
ResetDbType	640
ResetOracleDbType	640
ToString	641
OracleParameterCollection Class	641
OracleParameterCollection Members	643
OracleParameterCollection Static Methods	645
OracleParameterCollection Properties	645
Count	646
Item	646
Item[int]	647
Item[string]	648
IsFixedSize	648
IsReadOnly	649
IsSynchronized	649
SyncRoot	650
OracleParameterCollection Public Methods	650
Add	651
Add(object)	652
Add(OracleParameter)	653
Add(string, object)	653
Add(string, OracleDbType)	654
Add(string, OracleDbType, ParameterDirection)	655
Add(string, OracleDbType, object, ParameterDirection)	655
Add(string, OracleDbType, int, object, ParameterDirection)	656
Add(string, OracleDbType, int)	657
Add (string, OracleDbType, int, string)	658
Add(string, OracleDbType, int, ParameterDirection, bool, byte, byte, string, DataRowVersion, object)	659
AddRange	660
Clear	661
Contains	661
Contains(object)	662
Contains(string)	663
CopyTo	664
GetEnumerator	665
IndexOf	665
IndexOf(object)	666
IndexOf(String)	666
Insert	667
Remove	668
RemoveAt	669

RemoveAt(int)	669
RemoveAt(String)	670
OraclePermission Class	670
OraclePermission Members	671
OraclePermission Constructor	673
OraclePermission Static Methods	673
OraclePermission Public Properties	674
OraclePermission Public Methods	674
Add	675
Copy	676
IsSubsetOf	677
OraclePermissionAttribute Class	678
OraclePermissionAttribute Members	679
OraclePermissionAttribute Constructor	680
OraclePermissionAttribute Static Methods	681
OraclePermissionAttribute Public Properties	681
OraclePermissionAttribute Public Methods	682
CreatePermission	682
OracleRefreshAccessTokenEventArgs Class	683
OracleRefreshAccessTokenEventArgs Members	683
OracleRefreshAccessTokenEventArgs Properties	684
DbToken	684
PrivateKey	685
Token	685
OracleRoleCollection Class	686
OracleRoleCollection Members	687
OracleRoleCollection Methods	687
Add	688
Add(string)	688
Add(string, OracleOpaqueString)	690
Clear	691
Contains	691
Remove	692
OracleRoleCollection Properties	693
Count	693
Item	694
UseAllExcept	695
OracleRowUpdatedEventArgs Class	695
OracleRowUpdatedEventArgs Members	696
OracleRowUpdatedEventArgs Constructor	698
OracleRowUpdatedEventArgs Static Methods	698
OracleRowUpdatedEventArgs Properties	699

Command	699
OracleRowUpdatedEventArgs Public Methods	700
OracleRowUpdatedEventHandler Delegate	700
OracleRowUpdatingEventArgs Class	701
OracleRowUpdatingEventArgs Members	702
OracleRowUpdatingEventArgs Constructor	704
OracleRowUpdatingEventArgs Static Methods	704
OracleRowUpdatingEventArgs Properties	705
Command	705
OracleRowUpdatingEventArgs Public Methods	706
OracleRowUpdatingEventHandler Delegate	706
OracleSessionlessTransactionOptions Class	707
OracleSessionlessTransactionOptions Members	708
OracleSessionlessTransactionOptions Constructors	709
OracleSessionlessTransactionOptions()	709
OracleSessionlessTransactionOptions(int, OracleSessionlessTransactionStartBehavior, OracleSessionlessTransactionStartBehavior, bool)	710
OracleSessionlessTransactionOptions Properties	711
ResumeBehavior	711
StartBehavior	712
SuspendAfterExecution	712
Timeout	713
OracleShardingKey Class	713
OracleShardingKey Members	715
OracleShardingKey Constructors	715
OracleShardingKey()	716
OracleShardingKey(OracleDbType, object)	716
OracleShardingKey Instance Methods	717
SetShardingKey(OracleDbType, object)	717
Dispose	718
OracleSparseVector Class	718
OracleSparseVector Members	719
OracleSparseVector Constructors	721
OracleSparseVector(sparseVectorString)	721
OracleSparseVector(sparseVectorString, providerType)	722
OracleSparseVector(dimensions, indices)	723
OracleSparseVector(dimensions, indices, sparseValue)	724
OracleSparseVector(dimensions, indices, sparseValue)	725
OracleSparseVector(dimensions, indices, sparseValue)	725
OracleSparseVector Properties	726
Dimensions	727

VectorType	727
ProviderType	728
Value	729
Indices	729
Item	730
IsNull	731
OracleSparseVector Instance Methods	731
ToInt16Array()	732
ToFloatArray()	733
ToDoubleArray()	733
ToNonZeroInt16Array()	734
ToNonZeroFloatArray()	734
ToNonZeroDoubleArray()	735
ToString()	735
Dispose()	736
OracleTransaction Class	736
OracleTransaction Members	739
OracleTransaction Static Methods	740
OracleTransaction Properties	740
IsolationLevel	741
Connection	741
OracleTransaction Public Methods	742
Commit	743
Dispose	745
Rollback	745
Rollback()	746
Rollback(string)	747
Save	748
OracleVector Class	750
OracleVector Members	751
OracleVector Constructors	752
OracleVector(vectorString)	753
OracleVector(vectorString, providerType)	754
OracleVector(vectorValues)	754
OracleVector(vectorValues)	755
OracleVector(vectorValues)	756
OracleVector(vectorValues)	756
OracleVector Properties	757
Dimensions	757
VectorType	758
ProviderType	759
Value	759

Item[long]	760
OracleVector Instance Methods	761
ToInt16Array()	761
ToFloatArray()	762
ToDoubleArray()	763
ToByteArray()	763
ToString()	764
Dispose()	764
OracleAllowedLogonVersionClient Enumeration	765
OracleCollectionType Enumeration	766
OracleConnectionOpenReason Enumeration	766
OracleConnectionType Enumeration	767
OracleDBAPrivilege Enumeration	768
OracleDBShutdownMode Enumeration	769
OracleDBStartupMode Enumeration	770
OracleDbType Enumeration	771
OracleDRCPYPurity Enumeration	772
OracleIdentityType Enumeration	773
OracleParameterStatus Enumeration	774
OraclePasswordAuth Enumeration	775
OracleSessionlessTransactionStartBehavior Enumeration	776
OracleTokenAuth Enumeration	776
OracleVectorType Enumeration	778

7 Oracle Data Provider for .NET XML-Related Classes

OracleXmlCommandType Enumeration	1
OracleXmlQueryProperties Class	2
OracleXmlQueryProperties Members	5
OracleXmlQueryProperties Constructor	6
OracleXmlQueryProperties Properties	7
MaxRows	7
RootTag	8
RowTag	8
Xslt	9
XsltParams	9
OracleXmlQueryProperties Public Methods	10
Clone	10
OracleXmlSaveProperties Class	11
OracleXmlSaveProperties Members	14
OracleXmlSaveProperties Constructor	15
OracleXmlSaveProperties Properties	15

KeyColumnsList	16
RowTag	16
Table	17
UpdateColumnsList	17
Xslt	18
XsltParams	19
OracleXmlSaveProperties Public Methods	19
Clone	19
OracleXmlStream Class	20
OracleXmlStream Members	21
OracleXmlStream Constructor	23
OracleXmlStream Static Methods	23
OracleXmlStream Instance Properties	24
CanRead	24
CanSeek	25
CanWrite	25
Connection	26
Length	26
Position	27
Value	28
OracleXmlStream Instance Methods	28
Clone	29
Close	30
Dispose	30
Flush	31
Read	31
Read(byte[], int, int)	32
Read(char[], int, int)	32
ReadAsync	33
ReadAsync(byte[], int, int, CancellationToken)	34
ReadAsync(char[], int, int)	35
ReadAsync(char[], int, int, CancellationToken)	36
ReadAsync(byte[], int, int)	37
Seek	38
SetLength	39
Write	39
WriteLine	39
OracleXmlType Class	39
OracleXmlType Members	40
OracleXmlType Constructors	42
OracleXmlType(OracleClob)	43
OracleXmlType(OracleConnection, string)	44

OracleXmlType(OracleConnection, XmlReader)	44
OracleXmlType(OracleConnection, XmlDocument)	45
OracleXmlType Static Methods	46
OracleXmlType Static Fields	46
Null	47
OracleXmlType Instance Properties	47
Connection	48
IsEmpty	48
IsFragment	49
IsNull	49
IsSchemaBased	50
RootElement	51
Schema	51
SchemaUrl	52
Value	52
OracleXmlType Instance Methods	53
Clone	54
Dispose	54
Extract	55
Extract(string, string)	55
Extract(string, XmlNameSpaceManager)	56
GetStream	57
GetXmlDocument	57
GetXmlReader	58
Exists	59
Exists(string, string)	59
Exists(string, XmlNameSpaceManager)	60
Transform	61
Transform(OracleXmlType, string)	61
Transform(string, string)	62
Update	63
Update(string, string, string)	63
Update(string, XmlNameSpaceManager, string)	64
Update(string, string, OracleXmlType)	65
Update(string, XmlNameSpaceManager, OracleXmlType)	66
Validate	67

8 Oracle Data Provider for .NET HA Event Classes

OracleHAEventArgs Class	1
OracleHAEventArgs Members	2
OracleHAEventArgs Properties	3

DatabaseDomainName	3
DatabaseName	4
DrainTimeout	4
HostName	5
InstanceName	5
Reason	6
ServiceName	6
Source	7
Status	7
Time	8
OracleHAEventHandler Delegate	8
OracleHAEventSource Enumeration	9
OracleHAEventStatus Enumeration	10

9 Continuous Query Notification Classes

OracleDependency Class	1
OracleDependency Members	2
OracleDependency Constructors	4
OracleDependency ()	5
OracleDependency(OracleCommand)	5
OracleDependency(OracleCommand, bool, int, bool)	6
OracleDependency Static Fields	8
Address	8
Port	9
OracleDependency Static Methods	10
GetOracleDependency	10
OracleDependency Properties	11
DataSource	11
HasChanges	12
Id	13
IsEnabled	13
QueryBasedNotification	14
RegisteredQueryIDs	14
RegisteredResources	15
RowidInfo	16
UserName	16
OracleDependency Methods	17
AddCommandDependency	18
RemoveRegistration	19
OracleDependency Events	20
OnChange	20

OracleNotificationRequest Class	20
OracleNotificationRequest Members	21
OracleNotificationRequest Static Methods	22
OracleNotificationRequest Properties	23
IsNotifiedOnce	23
IsPersistent	24
Timeout	25
GroupingNotificationEnabled	25
GroupingType	26
GroupingInterval	27
OracleNotificationRequest Methods	27
OracleNotificationEventArgs Class	28
OracleNotificationEventArgs Members	29
OracleNotificationEventArgs Static Fields	30
OracleNotificationEventArgs Static Methods	30
OracleNotificationEventArgs Properties	31
Details	31
Info	32
ResourceNames	33
Source	33
Type	35
OracleNotificationEventArgs Methods	35
OnChangeEventHandler Delegate	36
OracleRowidInfo Enumeration	37
OracleNotificationType Enumeration	37
OracleNotificationSource Enumeration	38
OracleNotificationInfo Enumeration	39

10 Oracle Data Provider for .NET Globalization Classes

OracleGlobalization Class	1
OracleGlobalization Members	2
OracleGlobalization Static Methods	4
GetClientInfo	4
GetClientInfo()	5
GetClientInfo(OracleGlobalization)	6
GetThreadInfo	7
GetThreadInfo()	7
GetThreadInfo(OracleGlobalization)	8
SetThreadInfo	9
OracleGlobalization Properties	10
Calendar	11

ClientCharacterSet	12
Comparison	12
Currency	13
DateFormat	13
DateLanguage	14
DualCurrency	14
ISOCurrency	15
Language	16
LengthSemantics	16
NCharConversionException	17
NumericCharacters	18
Sort	18
Territory	19
TimeStampFormat	19
TimeStampTZFormat	20
TimeZone	20
OracleGlobalization Public Methods	21
Clone	22
Dispose	22
OracleDatabaseCharset Enumeration	23
OracleDatabaseNCharset Enumeration	29

11 Oracle Data Provider for .NET Transparent Application Failover Classes

OracleFailoverEventArgs Class	1
OracleFailoverEventArgs Members	3
OracleFailoverEventArgs Static Methods	4
OracleFailoverEventArgs Properties	5
FailoverEvent	5
FailoverType	6
OracleFailoverEvent	6
OracleFailoverType	7
OracleFailoverEventArgs Public Methods	7
OracleTAFMode Class	8
OracleTAFMode Constructors	8
OracleFailoverEventHandler Delegate	9
FailoverEvent Enumeration	10
FailoverReturnCode Enumeration	12
FailoverType Enumeration	12
OracleFailoverEvent Enumeration	13
OracleFailoverRestore Enumeration	14
OracleFailoverReturnCode Enumeration	16

12 Oracle Database Advanced Queuing and Transactional Event Queues Classes

OracleAQAgent Class	1
OracleAQAgent Members	2
OracleAQAgent Constructors	3
OracleAQAgent (string)	3
OracleAQAgent (string, string)	4
OracleAQAgent Properties	5
Address	5
Name	6
OracleAQDequeueOptions Class	6
OracleAQDequeueOptions Members	7
OracleAQDequeueOptions Constructor	8
OracleAQDequeueOptions Properties	9
ConsumerName	9
Correlation	10
DeliveryMode	10
DequeueMode	11
MessageId	12
NavigationMode	12
ProviderSpecificType	13
Visibility	13
Wait	14
OracleAQDequeueOptions Public Methods	15
Clone	15
OracleAQEnqueueOptions Class	16
OracleAQEnqueueOptions Members	17
OracleAQEnqueueOptions Constructor	17
OracleAQEnqueueOptions Properties	18
DeliveryMode	18
Visibility	19
OracleAQEnqueueOptions Public Methods	19
Clone	20
OracleAQMessage Class	20
OracleAQMessage Members	22
OracleAQMessage Constructors	23
OracleAQMessage()	23
OracleAQMessage(Object)	23
OracleAQMessage Properties	24

Correlation	25
Delay	26
DeliveryMode	26
DequeueAttempts	27
EnqueueTime	27
ExceptionQueue	28
Expiration	28
MessageId	29
OriginalMessageId	30
Payload	30
Priority	31
Recipients	31
SenderId	32
State	32
TransactionGroup	33
OracleAQMessageAvailableEventArgs Class	33
OracleAQMessageAvailableEventArgs Members	34
OracleAQMessageAvailableEventArgs Constructor	36
OracleAQMessageAvailableEventArgs Properties	36
AvailableMessages	37
ConsumerName	37
Correlation	38
Delay	38
DeliveryMode	39
EnqueueTime	39
ExceptionQueue	40
Expiration	40
MessageId	41
NotificationType	41
OriginalMessageId	42
Payload	42
Priority	43
Queue	43
QueueName	44
SenderId	44
State	45
OracleAQMessageAvailableEventHandler Delegate	45
OracleAQQueue Class	46
OracleAQQueue Members	47
OracleAQQueue Properties	49
AutoCommitTEQNotification	49
Connection	50

DequeueOptions	51
EnqueueOptions	51
MessageType	52
Name	53
Notification	53
NotificationConsumers	54
UdtTypeName	55
OracleAQQueue Constructors	55
OracleAQQueue(string)	56
OracleAQQueue(string, OracleConnection)	56
OracleAQQueue(string, OracleConnection, OracleAQMessageType)	57
OracleAQQueue(string, OracleConnection, OracleAQMessageType, string)	58
OracleAQQueue Static Methods	59
Listen	59
Listen(OracleConnection, OracleAQAgent[])	60
Listen(OracleConnection, OracleAQAgent[], int)	61
OracleAQQueue Public Methods	62
AcknowledgeTxEventQNotification	62
Dequeue	65
Dequeue()	66
Dequeue(OracleAQDequeueOptions)	66
DequeueArray	67
DequeueArray(int)	68
DequeueArray(int, OracleAQDequeueOptions)	69
Dispose	70
Enqueue	71
Enqueue(OracleAQMessage)	71
Enqueue(OracleAQMessage, OracleAQEnqueueOptions)	72
EnqueueArray	73
EnqueueArray(OracleAQMessage[])	73
EnqueueArray(OracleAQMessage[], OracleAQEnqueueOptions)	74
Listen	76
Listen(string[])	76
Listen (string[], int)	80
OracleAQQueue Events	81
MessageAvailable Event	81
OracleAQDequeueMode Enumeration	85
OracleAQMessageDeliveryMode Enumeration	86
OracleAQMessageState Enumeration	88
OracleAQMessageType Enumeration	88
OracleAQNavigationMode Enumeration	89
OracleAQNotificationGroupingType Enumeration	91

OracleAQNotificationType Enumeration	91
OracleAQVisibilityMode Enumeration	92
OracleTxEventQNotificationDirective Enumeration	93

13 Oracle Data Provider for .NET Types Classes

OracleBFile Class	1
OracleBFile Members	3
OracleBFile Constructors	6
OracleBFile(OracleConnection)	6
OracleBFile(OracleConnection, string, string)	7
OracleBFile Static Fields	8
MaxSize	8
Null	8
OracleBFile Static Methods	9
OracleBFile Instance Properties	9
CanRead	10
CanSeek	10
CanWrite	11
Connection	11
DirectoryName	12
FileExists	12
FileName	13
IsEmpty	14
IsNull	14
IsOpen	15
Length	15
Position	16
Value	16
OracleBFile Instance Methods	17
Clone	18
Close	20
CloseFile	21
Compare	21
CopyTo	23
CopyTo(OracleBlob)	24
CopyTo(OracleBlob, Int64)	24
CopyTo(Int64, OracleBlob, Int64, Int64)	25
CopyTo(OracleClob)	26
CopyTo(OracleClob, Int64)	27
CopyTo(Int64, OracleClob, Int64, Int64)	28
CopyToAsync	29

CopyToAsync(Int64, OracleBlob, Int64, Int64)	32
CopyToAsync(Int64, OracleBlob, Int64, Int64, CancellationToken)	33
CopyToAsync(OracleBlob, Int64)	34
CopyToAsync(OracleBlob, Int64, CancellationToken)	34
CopyToAsync(OracleBlob)	35
CopyToAsync(OracleBlob, CancellationToken)	36
CopyToAsync(Int64, OracleClob, Int64, Int64)	36
CopyToAsync(Int64, OracleClob, Int64, Int64, CancellationToken)	37
CopyToAsync(OracleClob, Int64)	38
CopyToAsync(OracleClob, Int64, CancellationToken)	39
CopyToAsync(OracleClob)	40
CopyToAsync(OracleClob, CancellationToken)	40
Dispose	41
Flush	42
FlushAsync	42
IsEqual	42
OpenFile	43
Read	44
ReadAsync	45
ReadAsync(byte[], int, int)	46
ReadAsync(byte[], int, int, CancellationToken)	47
Search	48
Seek	50
SetLength	52
Write	52
OracleBlob Class	53
OracleBlob Members	55
OracleBlob Constructors	58
OracleBlob(OracleConnection)	58
OracleBlob(OracleConnection, bool)	59
OracleBlob Static Fields	59
MaxSize	60
Null	60
OracleBlob Static Methods	61
OracleBlob Instance Properties	61
CanRead	62
CanSeek	62
CanWrite	63
Connection	63
IsEmpty	64
IsInChunkWriteMode	64
IsNull	65

IsTemporary	65
Length	66
OptimumChunkSize	66
Position	67
Value	67
OracleBlob Instance Methods	68
Append	69
Append(OracleBlob)	70
Append(byte[], int, int)	71
BeginChunkWrite	72
Clone	73
Close	74
Compare	74
CopyTo	75
CopyTo(OracleBlob)	76
CopyTo(OracleBlob, Int64)	77
CopyTo(Int64, OracleBlob, Int64, Int64)	78
CopyToAsync	79
CopyToAsync(Int64, OracleBlob, Int64, Int64)	81
CopyToAsync(Int64, OracleBlob, Int64, Int64, CancellationToken)	82
CopyToAsync(OracleBlob, Int64)	83
CopyToAsync(OracleBlob, Int64, CancellationToken)	84
CopyToAsync(OracleBlob)	85
CopyToAsync(OracleBlob, CancellationToken)	85
Dispose	86
EndChunkWrite	87
Erase	87
Erase()	88
Erase(Int64, Int64)	88
Flush	89
IsEqual	89
Read	90
ReadAsync	92
ReadAsync(byte[], int, int)	92
ReadAsync(byte[], int, int, CancellationToken)	93
Search	95
Seek	96
SetLength	97
Write	98
WriteAsync	100
WriteAsync(byte[], int, int)	100
WriteAsync(byte[], int, int, CancellationToken)	101

OracleClob Class	103
OracleClob Members	105
OracleClob Constructors	108
OracleClob(OracleConnection)	108
OracleClob(OracleConnection, bool, bool)	109
OracleClob Static Fields	110
MaxSize	110
Null	111
OracleClob Static Methods	111
OracleClob Instance Properties	111
CanRead	112
CanSeek	113
CanWrite	113
Connection	113
IsEmpty	114
IsInChunkWriteMode	114
IsNClob	115
IsNull	115
IsTemporary	116
Length	116
OptimumChunkSize	117
Position	117
Value	118
OracleClob Instance Methods	119
Append	120
Append(OracleClob)	121
Append(byte [], int, int)	121
Append(char [], int, int)	122
BeginChunkWrite	123
Clone	124
Close	125
Compare	126
CopyTo	127
CopyTo(OracleClob)	127
CopyTo(OracleClob, Int64)	128
CopyTo(Int64, OracleClob, Int64, Int64)	129
CopyToAsync	131
CopyToAsync(Int64, OracleClob, Int64, Int64)	132
CopyToAsync(Int64, OracleClob, Int64, Int64, CancellationToken)	133
CopyToAsync(OracleClob, Int64)	134
CopyToAsync(OracleClob, Int64, CancellationToken)	135
CopyToAsync(OracleClob)	136

CopyToAsync(OracleClob, CancellationToken)	137
Dispose	137
EndChunkWrite	138
Erase	139
Erase()	139
Erase(Int64, Int64)	139
Flush	140
GetHashCode	140
IsEqual	141
Read	141
Read(byte [], int, int)	142
Read(char [], int, int)	143
ReadAsync	145
ReadAsync(byte[], int, int, CancellationToken)	146
ReadAsync(byte[], int, int)	147
ReadAsync(char[], int, int)	148
ReadAsync(char[], int, int, CancellationToken)	149
Search	150
Search(byte[], Int64, Int64)	151
Search(char[], Int64, Int64)	152
Seek	153
SetLength	154
Write	155
WriteAsync	155
WriteAsync(byte[], int, int)	157
WriteAsync(byte[], int, int, CancellationToken)	158
WriteAsync(char[], int, int)	159
WriteAsync(char[], int, int, CancellationToken)	160
Write(byte[], int, int)	161
Write(char[], int, int)	162
OracleRefCursor Class	163
OracleRefCursor Members	166
OracleRefCursor Static Methods	167
OracleRefCursor Static Fields	168
Null	168
OracleRefCursor Properties	168
Connection	169
FetchSize	170
IsNull	171
RowSize	171
RowsToFetchPerRoundTrip	172
OracleRefCursor Instance Methods	173

Dispose	173
GetDataReader	174

14 Oracle Data Provider for .NET Types Structures

OracleBinary Structure	1
OracleBinary Members	3
OracleBinary Constructor	5
OracleBinary Static Fields	5
Null	6
OracleBinary Static Methods	6
Concat	7
Equals	7
GetXsdType	8
GreaterThan	9
GreaterThanOrEqual	10
LessThan	10
LessThanOrEqual	11
NotEquals	12
OracleBinary Static Operators	13
operator +	13
operator ==	14
operator >	15
operator >=	16
operator !=	16
operator <	17
operator <=	18
OracleBinary Static Type Conversion Operators	18
explicit operator byte[]	19
implicit operator OracleBinary	19
OracleBinary Properties	20
IsNull	20
Item	21
Length	22
Value	22
OracleBinary Instance Methods	23
CompareTo	23
Equals	25
GetHashCode	25
ToString	26
OracleBoolean Structure	26
OracleBoolean Members	28

OracleBoolean Constructors	31
OracleBoolean(bool)	32
OracleBoolean(int)	32
OracleBoolean Static Fields	33
False	33
Null	34
One	34
True	34
Zero	35
OracleBoolean Static Methods	35
And	36
Equals	36
GreaterThan	37
GreaterThanOrEquals	38
LessThan	38
LessThanOrEquals	39
NotEquals	40
OnesComplement	40
Or	41
Parse	42
Xor	42
OracleBoolean Static Operators	43
operator >	44
operator >=	45
operator <	45
operator <=	46
operator ==	47
operator !=	47
operator !	48
operator ~	49
operator false	49
operator true	50
operator &	50
operator	51
operator ^	52
OracleBoolean Static Type Conversions	52
implicit operator OracleBoolean	53
explicit operator bool	53
explicit operator OracleBoolean	54
explicit operator OracleBoolean(byte)	55
explicit operator OracleBoolean(Decimal)	55
explicit operator OracleBoolean(Double)	56

explicit operator OracleBoolean(Int16)	56
explicit operator OracleBoolean(int)	57
explicit operator OracleBoolean(Int64)	57
explicit operator OracleBoolean(Single)	58
explicit operator OracleBoolean(String)	58
OracleBoolean Properties	59
ByteValue	59
IsFalse	59
IsNull	60
IsTrue	60
Value	61
OracleBoolean Instance Methods	61
CompareTo	62
Equals	63
GetHashCode	64
ToString	64
OracleDate Structure	65
OracleDate Members	66
OracleDate Constructors	69
OracleDate(DateTime)	69
OracleDate(string)	70
OracleDate(int, int, int)	71
OracleDate(int, int, int, int, int, int)	72
OracleDate(byte [])	73
OracleDate Static Fields	73
MaxValue	74
MinValue	74
Null	74
OracleDate Static Methods	75
Equals	75
GreaterThan	76
GreaterThanOrEqual	77
LessThan	78
LessThanOrEqual	78
NotEquals	79
GetSysDate	80
Parse	80
OracleDate Static Operators	82
operator ==	82
operator >	83
operator >=	84
operator !=	84

operator <	85
operator <=	86
OracleDate Static Type Conversions	87
explicit operator DateTime	87
explicit operator OracleDate	88
explicit operator OracleDate(DateTime)	88
explicit operator OracleDate(OracleTimeStamp)	89
explicit operator OracleDate(string)	89
OracleDate Properties	91
BinData	91
Day	92
IsNull	92
Hour	93
Minute	93
Month	94
Second	94
Value	95
Year	95
OracleDate Methods	96
CompareTo	96
Equals	97
GetHashCode	98
GetDaysBetween	98
ToOracleTimeStamp	99
ToString	99
OracleDecimal Structure	101
OracleDecimal Members	102
OracleDecimal Constructors	107
OracleDecimal(byte [])	108
OracleDecimal(decimal)	109
OracleDecimal(double)	109
OracleDecimal(int)	110
OracleDecimal(float)	111
OracleDecimal(long)	111
OracleDecimal(string)	112
OracleDecimal(string, string)	112
OracleDecimal Static Fields	114
MaxPrecision	114
MaxScale	115
MaxValue	115
MinScale	115
MinValue	116

NegativeOne	116
Null	116
One	117
Pi	117
Zero	117
OracleDecimal Static (Comparison) Methods	118
Equals	118
GreaterThan	119
GreaterThanOrEqual	120
LessThan	120
LessThanOrEqual	121
NotEquals	122
OracleDecimal Static (Manipulation) Methods	123
Abs	124
Add	124
AdjustScale	125
Ceiling	126
ConvertToPrecScale	127
Divide	128
Floor	129
Max	129
Min	130
Mod	130
Multiply	131
Negate	132
Parse	132
Round	133
SetPrecision	134
Shift	135
Sign	135
Sqrt	136
Subtract	137
Truncate	137
OracleDecimal Static (Logarithmic) Methods	138
Exp	138
Log	139
Log(OracleDecimal)	140
Log(OracleDecimal, int)	140
Log(OracleDecimal, OracleDecimal)	141
Pow	142
Pow(OracleDecimal, int)	142
Pow(OracleDecimal, OracleDecimal)	143

OracleDecimal Static (Trigonometric) Methods	144
Acos	144
Asin	145
Atan	145
Atan2	146
Cos	147
Sin	147
Tan	148
Cosh	149
Sinh	149
Tanh	150
OracleDecimal Static (Comparison) Operators	150
operator +	151
operator /	152
operator ==	152
operator >	153
operator >=	154
operator !=	154
operator <	155
operator <=	156
operator *	157
operator -	157
operator -	158
operator%	159
OracleDecimal Static Operators (Conversion from .NET Type to OracleDecimal)	159
implicit operator OracleDecimal	160
implicit operator OracleDecimal(decimal)	160
implicit operator OracleDecimal(int)	161
implicit operator OracleDecimal(long)	161
explicit operator OracleDecimal	162
explicit operator OracleDecimal(double)	162
explicit operator OracleDecimal(string)	163
OracleDecimal Static Operators (Conversion from OracleDecimal to .NET)	164
explicit operator byte	164
explicit operator decimal	165
explicit operator double	165
explicit operator short	166
explicit operator int	167
explicit operator long	167
explicit operator float	168
OracleDecimal Properties	168
BinData	169

Format	170
IsInt	170
IsNull	171
IsPositive	171
IsZero	172
Value	172
OracleDecimal Instance Methods	173
CompareTo	174
Equals	174
GetHashCode	175
ToByte	176
ToDouble	176
ToInt16	177
ToInt32	177
ToInt64	178
ToSingle	178
ToString	179
OracleIntervalDS Structure	180
OracleIntervalDS Members	181
OracleIntervalDS Constructors	184
OracleIntervalDS(TimeSpan)	185
OracleIntervalDS(string)	185
OracleIntervalDS(double)	186
OracleIntervalDS(int, int, int, int, double)	187
OracleIntervalDS(int, int, int, int, int)	188
OracleIntervalDS(byte[])	189
OracleIntervalDS Static Fields	189
MaxValue	190
MinValue	190
Null	191
Zero	191
OracleIntervalDS Static Methods	191
Equals	192
GreaterThan	193
GreaterThanOrEqual	194
LessThan	194
LessThanOrEqual	195
NotEquals	196
Parse	197
SetPrecision	198
OracleIntervalDS Static Operators	199
operator +	199

operator ==	200
operator >	201
operator >=	201
operator !=	202
operator <	203
operator <=	204
operator -	204
operator -	205
operator *	206
operator /	206
OracleIntervalDS Type Conversions	207
explicit operator TimeSpan	208
explicit operator OracleIntervalDS	208
implicit operator OracleIntervalDS	209
OracleIntervalDS Properties	210
BinData	210
Days	211
Hours	211
IsNull	212
Milliseconds	212
Minutes	213
Nanoseconds	213
Seconds	214
TotalDays	214
Value	215
OracleIntervalDS Methods	215
CompareTo	216
Equals	217
GetHashCode	217
ToString	218
OracleIntervalYM Structure	218
OracleIntervalYM Members	220
OracleIntervalYM Constructors	223
OracleIntervalYM(long)	223
OracleIntervalYM(string)	224
OracleIntervalYM(double)	225
OracleIntervalYM(int, int)	225
OracleIntervalYM(byte[])	226
OracleIntervalYM Static Fields	227
MaxValue	227
MinValue	228
Null	228

Zero	228
OracleIntervalYM Static Methods	229
Equals	229
GreaterThan	230
GreaterThanOrEqual	231
LessThan	232
LessThanOrEqual	232
NotEquals	233
Parse	234
SetPrecision	235
OracleIntervalYM Static Operators	236
operator +	236
operator ==	237
operator >	238
operator >=	238
operator !=	239
operator <	240
operator <=	240
operator -	241
operator -	242
operator *	242
operator /	243
OracleIntervalYM Type Conversions	244
explicit operator long	244
explicit operator OracleIntervalYM	245
implicit operator OracleIntervalYM	246
OracleIntervalYM Properties	246
BinData	247
IsNull	247
Months	248
TotalYears	248
Value	249
Years	249
OracleIntervalYM Methods	250
CompareTo	250
Equals	251
GetHashCode	252
ToString	252
OracleString Structure	253
OracleString Members	254
OracleString Constructors	257
OracleString(string)	258

OracleString(string, bool)	258
OracleString(byte [], bool)	259
OracleString(byte [], bool, bool)	259
OracleString(byte [], int, int, bool)	260
OracleString(byte [], int, int, bool, bool)	261
OracleString Static Fields	262
Null	262
OracleString Static Methods	263
Concat	263
Equals	264
GreaterThan	265
GreaterThanOrEqual	265
LessThan	266
LessThanOrEqual	267
NotEquals	267
OracleString Static Operators	268
operator +	269
operator ==	269
operator >	270
operator >=	271
operator !=	271
operator <	272
operator <=	273
OracleString Type Conversions	274
explicit operator string	274
implicit operator OracleString	275
OracleString Properties	275
IsCaseIgnored	276
IsNull	277
Item	277
Length	277
Value	278
OracleString Methods	278
Clone	279
CompareTo	280
Equals	281
GetHashCode	281
GetNonUnicodeBytes	282
GetUnicodeBytes	282
ToString	283
OracleTimeStamp Structure	283
OracleTimeStamp Members	285

OracleTimeStamp Constructors	289
OracleTimeStamp(DateTime)	290
OracleTimeStamp(string)	290
OracleTimeStamp(int, int, int)	291
OracleTimeStamp(int, int, int, int, int, int)	292
OracleTimeStamp(int, int, int, int, int, int, double)	293
OracleTimeStamp(int, int, int, int, int, int, int)	294
OracleTimeStamp(byte [])	295
OracleTimeStamp Static Fields	295
MaxValue	296
MinValue	296
Null	297
OracleTimeStamp Static Methods	297
Equals	298
GreaterThan	299
GreaterThanOrEqual	299
LessThan	300
LessThanOrEqual	301
NotEquals	301
GetSysDate	302
Parse	303
SetPrecision	304
OracleTimeStamp Static Operators	305
operator +	306
operator + (OracleTimeStamp, OracleIntervalDS)	306
operator + (OracleTimeStamp, OracleIntervalYM)	307
operator + (OracleTimeStamp, TimeSpan)	308
operator ==	308
operator >	309
operator >=	310
operator !=	310
operator <	311
operator <=	312
operator -	313
operator - (OracleTimeStamp, OracleIntervalDS)	313
operator - (OracleTimeStamp, OracleIntervalYM)	314
operator - (OracleTimeStamp, TimeSpan)	315
OracleTimeStamp Static Type Conversions	315
explicit operator OracleTimeStamp	316
explicit operator OracleTimeStamp(OracleTimeStampLTZ)	316
explicit operator OracleTimeStamp(OracleTimeStampTZ)	317
explicit operator OracleTimeStamp(string)	318

implicit operator OracleTimeStamp	319
implicit operator OracleTimeStamp(OracleDate)	319
implicit operator OracleTimeStamp(DateTime)	320
explicit operator DateTime	321
OracleTimeStamp Properties	321
BinData	322
Day	322
IsNull	323
Hour	323
Millisecond	324
Minute	324
Month	325
Nanosecond	325
Second	326
Value	326
Year	327
OracleTimeStamp Methods	327
AddDays	328
AddHours	329
AddMilliseconds	329
AddMinutes	330
AddMonths	331
AddNanoseconds	331
AddSeconds	332
AddYears	332
CompareTo	333
Equals	334
GetHashCode	335
GetDaysBetween	335
GetYearsBetween	336
ToOracleDate	336
ToOracleTimeStampLTZ	337
ToOracleTimeStampTZ	338
ToString	338
OracleTimeStampLTZ Structure	339
OracleTimeStampLTZ Members	341
OracleTimeStampLTZ Constructors	345
OracleTimeStampLTZ(DateTime)	346
OracleTimeStampLTZ(string)	346
OracleTimeStampLTZ(int, int, int)	348
OracleTimeStampLTZ(int, int, int, int, int, int)	349
OracleTimeStampLTZ(int, int, int, int, int, int, double)	349

OracleTimeStampLTZ(int, int, int, int, int, int, int)	350
OracleTimeStampLTZ(byte [])	351
OracleTimeStampLTZ Static Fields	352
MaxValue	353
MinValue	353
Null	353
OracleTimeStampLTZ Static Methods	354
Equals	354
GetLocalTimeZoneName	355
GetLocalTimeZoneOffset	356
GetSysDate	356
GreaterThan	357
GreaterThanOrEqual	357
LessThan	358
LessThanOrEqual	359
NotEquals	359
Parse	360
SetPrecision	362
OracleTimeStampLTZ Static Operators	362
operator +	363
operator + (OracleTimeStampLTZ, OracleIntervalDS)	364
operator + (OracleTimeStampLTZ, OracleIntervalYM)	364
operator + (OracleTimeStampLTZ, TimeSpan)	365
operator ==	366
operator >	366
operator >=	367
operator !=	368
operator <	369
operator <=	369
operator -	370
operator - (OracleTimeStampLTZ, OracleIntervalDS)	371
operator - (OracleTimeStampLTZ, OracleIntervalYM)	371
operator - (OracleTimeStampLTZ, TimeSpan)	372
OracleTimeStampLTZ Static Type Conversions	373
explicit operator OracleTimeStampLTZ	373
explicit operator OracleTimeStampLTZ(OracleTimeStamp)	374
explicit operator OracleTimeStampLTZ(OracleTimeStampTZ)	374
explicit operator OracleTimeStampLTZ(string)	375
implicit operator OracleTimeStampLTZ	376
implicit operator OracleTimeStampLTZ(OracleDate)	377
implicit operator OracleTimeStampLTZ(DateTime)	377
explicit operator DateTime	378

OracleTimeStampLTZ Properties	379
BinData	379
Day	380
IsNull	380
Hour	381
Millisecond	381
Minute	382
Month	382
Nanosecond	383
Second	383
Value	384
Year	384
OracleTimeStampLTZ Methods	385
AddDays	386
AddHours	386
AddMilliseconds	387
AddMinutes	388
AddMonths	388
AddNanoseconds	389
AddSeconds	389
AddYears	390
CompareTo	391
Equals	392
GetHashCode	392
GetDaysBetween	393
GetYearsBetween	393
ToOracleDate	394
ToOracleTimeStamp	394
ToOracleTimeStampTZ	395
ToString	396
ToUniversalTime	397
OracleTimeStampTZ Structure	397
OracleTimeStampTZ Members	399
OracleTimeStampTZ Constructors	403
OracleTimeStampTZ(DateTime)	404
OracleTimeStampTZ(DateTime, string)	405
OracleTimeStampTZ(string)	406
OracleTimeStampTZ(int, int, int)	407
OracleTimeStampTZ(int, int, int, string)	408
OracleTimeStampTZ(int, int, int, int, int, int)	409
OracleTimeStampTZ(int, int, int, int, int, int, string)	410
OracleTimeStampTZ(int, int, int, int, int, int, double)	411

OracleTimeStampTZ(int, int, int, int, int, int, double, string)	412
OracleTimeStampTZ(int, int, int, int, int, int, int)	413
OracleTimeStampTZ(int, int, int, int, int, int, int, string)	414
OracleTimeStampTZ(byte [])	416
OracleTimeStampTZ Static Fields	416
MaxValue	417
MinValue	417
Null	418
OracleTimeStampTZ Static Methods	418
Equals	419
GetSysDate	419
GreaterThan	420
GreaterThanOrEqual	421
LessThan	421
LessThanOrEqual	422
NotEquals	423
Parse	424
SetPrecision	425
OracleTimeStampTZ Static Operators	426
operator +	427
operator +(OracleTimeStampTZ, OracleIntervalDS)	427
operator +(OracleTimeStampTZ, OracleIntervalYM)	428
operator +(OracleTimeStampTZ, TimeSpan)	429
operator ==	429
operator >	430
operator >=	431
operator !=	431
operator <	432
operator <=	433
operator -	434
operator - (OracleTimeStampTZ, OracleIntervalDS)	434
operator - (OracleTimeStampTZ, OracleIntervalYM)	435
operator - (OracleTimeStampTZ value1, TimeSpan value2)	436
OracleTimeStampTZ Static Type Conversions	436
explicit operator OracleTimeStampTZ	437
explicit operator OracleTimeStampTZ(OracleTimeStamp)	437
explicit operator OracleTimeStampTZ(OracleTimeStampLTZ)	438
explicit operator OracleTimeStampTZ(string)	439
implicit operator OracleTimeStampTZ	440
implicit operator OracleTimeStampTZ(OracleDate)	441
implicit operator OracleTimeStampTZ(DateTime)	441
explicit operator DateTime	442

OracleTimeStampTZ Properties	443
BinData	444
Day	444
IsNull	445
Hour	445
Millisecond	445
Minute	446
Month	446
Nanosecond	447
Second	447
TimeZone	448
Value	448
Year	449
OracleTimeStampTZ Methods	449
AddDays	450
AddHours	451
AddMilliseconds	452
AddMinutes	452
AddMonths	453
AddNanoseconds	454
AddSeconds	454
AddYears	455
CompareTo	455
Equals	456
GetDaysBetween	457
GetHashCode	458
GetTimeZoneOffset	458
GetYearsBetween	458
ToLocalTime	459
ToOracleDate	460
ToOracleTimeStampLTZ	460
ToOracleTimeStamp	461
ToString	461
ToUniversalTime	463
INullable Interface	463
INullable Interface Members	464
INullable Interface Properties	464
IsNull	465

15 Oracle Data Provider for .NET Types Exceptions

OracleTypeException Class	1
OracleTypeException Members	2
OracleTypeException Constructors	3
OracleTypeException(string)	4
OracleTypeException(SerializationInfo, StreamingContext)	4
OracleTypeException Static Methods	5
OracleTypeException Properties	5
Message	6
Number	6
Source	7
OracleTypeException Methods	7
ToString	7
OracleNullValueException Class	8
OracleNullValueException Members	9
OracleNullValueException Constructors	10
OracleNullValueException()	11
OracleNullValueException(string)	11
OracleNullValueException Static Methods	12
OracleNullValueException Properties	12
OracleNullValueException Methods	12
OracleTruncateException Class	13
OracleTruncateException Members	14
OracleTruncateException Constructors	15
OracleTruncateException()	15
OracleTruncateException(string)	16
OracleTruncateException Static Methods	16
OracleTruncateException Properties	17
OracleTruncateException Methods	17

16 Oracle Data Provider for .NET UDT-Related Classes

OracleCustomTypeMappingAttribute Class	1
OracleCustomTypeMappingAttribute Members	3
OracleCustomTypeMappingAttribute Constructors	4
OracleCustomTypeMappingAttribute(string)	4
OracleCustomTypeMappingAttribute Static Methods	5
OracleCustomTypeMappingAttribute Properties	5
UdtTypeName	6
OracleCustomTypeMappingAttribute Methods	6
OracleObjectMappingAttribute Class	7

OracleObjectMappingAttribute Members	8
OracleObjectMappingAttribute Constructors	9
OracleObjectMappingAttribute(string)	10
OracleObjectMappingAttribute(int)	10
OracleObjectMappingAttribute Static Methods	11
OracleObjectMappingAttribute Properties	11
AttributeIndex	12
AttributeName	12
OracleObjectMappingAttribute Methods	13
OracleArrayMappingAttribute Class	13
OracleArrayMappingAttribute Members	14
OracleArrayMappingAttribute Constructors	15
OracleArrayMappingAttribute()	16
OracleArrayMappingAttribute Static Methods	16
OracleArrayMappingAttribute Properties	17
OracleArrayMappingAttribute Methods	17
IOracleCustomType Interface	18
IOracleCustomType Members	18
IOracleCustomType Interface Methods	19
FromCustomObject(OracleConnection, IntPtr)	20
FromCustomObject(OracleConnection, object)	20
ToCustomObject(OracleConnection, IntPtr)	21
ToCustomObject(OracleConnection, object)	22
IOracleCustomTypeFactory Interface	23
IOracleCustomTypeFactory Members	23
IOracleCustomTypeFactory Interface Methods	24
CreateObject	24
IOracleArrayTypeFactory Interface	25
IOracleArrayTypeFactory Members	25
IOracleArrayTypeFactory Interface Methods	26
CreateArray	26
CreateStatusArray	27
OracleUdt Class	28
OracleUdt Members	29
OracleUDT Static Methods	29
GetValue	30
GetValue(OracleConnection, IntPtr, string)	31
GetValue(OracleConnection, IntPtr, int)	32
GetValue(OracleConnection, IntPtr, string, out object)	33
GetValue(OracleConnection, IntPtr, int, out object)	34
GetValue(OracleConnection, object, string)	35
GetValue(OracleConnection, object, int)	37

GetValue(OracleConnection, object, string, out object)	38
GetValue(OracleConnection, object, int, out object)	39
IsDBNull	41
IsDBNull(OracleConnection, IntPtr, string)	41
IsDBNull(OracleConnection, IntPtr, int)	42
IsDBNull(OracleConnection, object, string)	43
IsDBNull(OracleConnection, object, int)	44
SetValue	45
SetValue(OracleConnection, IntPtr, string, object)	46
SetValue(OracleConnection, IntPtr, int, object)	47
SetValue(OracleConnection, IntPtr, string, object, object)	48
SetValue(OracleConnection, IntPtr, int, object, object)	49
SetValue(OracleConnection, object, string, object)	50
SetValue(OracleConnection, object, int, object)	51
SetValue(OracleConnection, object, string, object, object)	52
SetValue(OracleConnection, object, int, object, object)	53
OracleRef Class	54
OracleRef Members	55
OracleRef Constructors	57
OracleRef(OracleConnection, string)	58
OracleRef(OracleConnection, string, string)	58
OracleRef Static Fields	59
Null	60
OracleRef Static Methods	60
OracleRef Instance Properties	61
Connection	61
HasChanges	62
IsLocked	62
IsNull	63
ObjectTableName	63
Value	64
Oracle Ref Instance Methods	65
Clone	65
Delete	66
Dispose	67
Flush	67
GetCustomObject	68
GetCustomObject(OracleUdtFetchOption)	68
GetCustomObject(OracleUdtFetchOption, int)	70
GetCustomObjectForUpdate	71
GetCustomObjectForUpdate(bool)	72
GetCustomObjectForUpdate(bool, int)	73

IsEqual	74
Lock	74
Update	75
OracleUdtFetchOption Enumeration	77
OracleUdtStatus Enumeration	77

17 Oracle Data Provider for .NET Bulk Copy Classes

OracleBulkCopy Class	1
OracleBulkCopy Members	2
OracleBulkCopy Constructors	4
OracleBulkCopy(OracleConnection)	4
OracleBulkCopy(string)	5
OracleBulkCopy(OracleConnection, OracleBulkCopyOptions)	6
OracleBulkCopy(string, OracleBulkCopyOptions)	7
OracleBulkCopy Properties	7
BatchSize	8
BulkCopyOptions	9
BulkCopyTimeout	10
ColumnMappings	10
Connection	11
DestinationPartitionName	12
DestinationSchemaName	12
DestinationTableName	13
NotifyAfter	13
OracleBulkCopy Public Methods	14
Close	14
Dispose	15
WriteToServer	15
WriteToServer(DataRow[])	16
WriteToServer(DataTable)	17
WriteToServer(IDataReader)	17
WriteToServer(DataTable, DataRowState)	18
WriteToServer(OracleRefCursor)	19
OracleBulkCopy Events	20
OracleRowsCopied	20
OracleBulkCopyColumnMapping Class	21
OracleBulkCopyColumnMapping Members	22
OracleBulkCopyColumnMapping Constructors	23
OracleBulkCopyColumnMapping()	24
OracleBulkCopyColumnMapping(int, int)	24
OracleBulkCopyColumnMapping(int, string)	25

OracleBulkCopyColumnMapping(string, int)	25
OracleBulkCopyColumnMapping(string, string)	26
OracleBulkCopyColumnMapping Methods	26
CompareTo	27
OracleBulkCopyColumnMapping Properties	27
DestinationColumn	28
DestinationOrdinal	28
SourceColumn	29
SourceOrdinal	29
OracleBulkCopyColumnMappingCollection Class	30
OracleBulkCopyColumnMappingCollection Members	31
OracleBulkCopyColumnMappingCollection Properties	32
Item[index]	33
OracleBulkCopyColumnMappingCollection Public Methods	33
Add	34
Add(OracleBulkCopyColumnMapping)	35
Add(int, int)	35
Add(int, string)	36
Add(string, int)	37
Add(string, string)	38
Clear	39
Contains	40
CopyTo	40
IndexOf	41
Insert	41
Remove	42
RemoveAt	43
OracleBulkCopyOptions Enumeration	43
OracleRowsCopiedEventHandler Delegate	46
OracleRowsCopiedEventArgs Class	47
OracleRowsCopiedEventArgs Members	48
OracleRowsCopiedEventArgs Constructors	49
OracleRowsCopiedEventArgs(long)	49
OracleRowsCopiedEventArgs Properties	50
Abort	50
RowsCopied	51

18 Oracle Data Provider for .NET OpenTelemetry and Metrics Classes

OracleDataProviderInstrumentationOptions Class	1
OracleDataProviderInstrumentationOptions Properties	2
AddDBInfoToDisplayName	2

EnableConnectionLevelAttributes	3
EnableDBRoundTripTracing	4
EnableOpenCloseTracing	5
EnableSqlIdTracing	5
InstrumentOracleDataReaderRead	6
RecordException	6
RequireApplicationRootSpanCreation	7
SetDbStatementForStoredProcedure	8
SetDbStatementForText	8
AddOracleDataProviderInstrumentation	9
OraclePoolNameCollection Class	9
OraclePoolNameCollection Methods	10
Add(string, string)	11
Remove (string)	11

19 Oracle Data Provider for .NET Oracle Identity and Access Management Classes

OciTokenAuthentication Class	1
OciTokenAuthentication Members	2
OciTokenAuthentication Constructor()	3
OciTokenAuthentication Properties	3
Compartment	3
ConfigurationFile	4
Database	5
Profile	6

20 Oracle Data Provider for .NET Microsoft Entra ID Classes

AzureTokenAuthentication Class	1
AzureTokenAuthentication Members	2
AzureTokenAuthentication Constructors	3
AzureTokenAuthentication()	3
AzureTokenAuthentication(string)	4
AzureTokenAuthentication Properties	4
DatabaseApplicationIdUri	5
ClientId	5
TenantId	6
RedirectUri	6
ClientSecret	7
ClientCertificate	7
ClientCertificatePassword	8

21 Oracle Database Vector Store Connector Classes

OracleCollection Class	1
OracleCollection Constructors	2
OracleCollection<TKey, TRecord>(OracleDataSource, string, OracleCollectionOptions?)	3
OracleCollection<TKey, TRecord>(string, string, OracleCollectionOptions?)	4
OracleCollection Properties	4
Name	5
OracleCollection Public Methods	5
CollectionExistsAsync	6
DeleteAsync	6
DeleteAsync(TKey key, CancellationTokens)	7
DeleteAsync(IEnumerable<TKey> keys, CancellationTokens)	7
Dispose	8
EnsureCollectionDeletedAsync	8
EnsureCollectionExistsAsync	9
GetAsync	9
GetAsync(TKey , RecordRetrieveOptions?, CancellationTokens)	9
GetAsync(IEnumerable<TKey>, RecordRetrieveOptions?, CancellationTokens)	10
GetAsync(Expression<Func<TRecord, bool>>, int, FilteredRecordRetrievalOptions<TRecord>?, CancellationTokens)	11
GetService	12
SearchAsync	13
UpsertAsync	14
UpsertAsync(TRecord, CancellationTokens)	14
UpsertAsync(IEnumerable<TRecord>, CancellationTokens)	15
OracleCollectionOptions Class	16
OracleCollectionOptions Constructor	16
OracleCollectionOptions Properties	17
EmbeddingGenerator	17
Schema	18
Definition	18
OracleDynamicCollection Class	18
OracleDynamicCollection Constructors	19
OracleDynamicCollection(OracleDataSource, string, OracleCollectionOptions)	20
OracleDynamicCollection(string, string, OracleCollectionOptions)	21
OracleDynamicCollection Properties	22
Name	22
OracleDynamicCollection Public Methods	22
GetAsync	23

CollectionExistsAsync	23
DeleteAsync	24
DeleteAsync(Object key, CancellationTokens)	24
DeleteAsync(IEnumerable<Object> keys, CancellationTokens)	25
Dispose	26
EnsureCollectionDeletedAsync	26
EnsureCollectionExistsAsync	26
GetAsync(Object , RecordRetrieveOptions?, CancellationTokens)	27
GetAsync(IEnumerable<Object>, RecordRetrieveOptions?, CancellationTokens)	28
GetAsync(Expression<Func<Dictionary<string, object?>, bool>>, int,<Dictionary<string, object?>>?, CancellationTokens)	29
GetService	30
SearchAsync	30
UpsertAsync	32
UpsertAsync(Dictionary<string, object?>, CancellationTokens)	32
UpsertAsync(IEnumerable<Dictionary<string, object?>>, CancellationTokens)	33
OracleServiceCollectionExtensions Class	34
OracleServiceCollectionExtensions Static Methods	34
AddKeyedOracleCollection	35
AddKeyedOracleCollection<TKey, TRecord>(IServiceCollection, object?, string, Func<IServiceProvider, string>, Func<IServiceProvider, OracleVectorCollectionOptions>?, ServiceLifetime)	35
AddKeyedOracleCollection<TKey, TRecord>(IServiceCollection, object?, string, string,OracleCollectionOptions?, ServiceLifetime)	36
AddKeyedOracleVectorStore	37
AddKeyedOracleVectorStore(IServiceCollection, object?, Func<IServiceProvider, string>, Func<IServiceProvider, OracleVectorStoreOptions>?, ServiceLifetime)	37
AddKeyedOracleVectorStore(IServiceCollection, object?, string, OracleVectorStoreOptions?, ServiceLifetime)	38
AddOracleCollection	38
AddOracleCollection<TKey, TRecord>(IServiceCollection, string, Func<IServiceProvider, string>, Func<IServiceProvider, OracleCollectionsOptions>?, ServiceLifetime)	39
AddOracleCollection<TKey, TRecord>(IServiceCollection, string, string, OracleCollectionOptions?, ServiceLifetime)	40
AddOracleCollection<TKey, TRecord>(this IServiceCollection, string, OracleCollectionOptions?, ServiceLifetime)	40
AddOracleVectorStore	41
AddOracleVectorStore(this IServiceCollection, OracleVectorStoreOptions?, ServiceLifetime)	41
AddOracleVectorStore(IServiceCollection, string, OracleVectorStoreOptions?, ServiceLifetime)	42
AddOracleVectorStore(IServiceCollection, Func<IServiceProvider, string>, Func<IServiceProvider, OracleVectorStoreOptions>?, ServiceLifetime)	43
OracleVectorStore Class	43

OracleVectorStore Constructors	44
OracleVectorStore(OracleDataSource, OracleVectorStoreOptions?)	45
OracleVectorStore(string connectionString, OracleVectorStoreOptions?)	45
OracleVectorStore Public Methods	46
CollectionExistsAsync	47
Dispose	47
EnsureCollectionDeletedAsync	48
GetCollection(string, VectorStoreCollectionDefinition?)	48
GetDynamicCollection(string, VectorStoreCollectionDefinition)	49
GetService	50
ListCollectionNamesAsync	51
OracleVectorStoreOptions Class	51
OracleVectorStoreOptions Constructor	52
OracleVectorStoreOptions Properties	52
EmbeddingGenerator	52
Schema	53

A Oracle Schema Collections

Common Schema Collections	A-1
MetaDataCollections	A-1
DataSourceInformation	A-2
DataTypes	A-3
Restrictions	A-5
ReservedWords	A-5
ODP.NET-Specific Schema Collection	A-6
Tables	A-6
Columns	A-6
Views	A-7
XMLSchema	A-7
Users	A-8
Synonyms	A-8
Sequences	A-9
Functions	A-9
Procedures	A-10
ProcedureParameters	A-11
Arguments	A-12
Packages	A-12
PackageBodies	A-13
JavaClasses	A-14
Indexes	A-14
IndexColumns	A-17

PrimaryKeys	A-17
ForeignKeys	A-18
ForeignKeyColumns	A-19
UniqueKeys	A-19

B Mapping LINQ Canonical Functions and Oracle Functions

Glossary

Index

List of Examples

2-1	<u>Setting the profile which could be used for all connections</u>	<u>29</u>
2-2	<u>Setting the Profile for a Specific Data Source</u>	<u>29</u>
2-3	<u>Setting the Profile for a Specific User Id</u>	<u>30</u>
2-4	<u>Setting the Profile for a Specific Data Source and User Id'</u>	<u>30</u>
2-5	<u>Configuring Multiple Default Profile Entries</u>	<u>30</u>
2-6	<u>Code Sample</u>	<u>47</u>
3-1	<u>Using Database Resident Connection Pooling: Sample Code</u>	<u>61</u>
3-2	<u>Using Pluggable Database: Sample Code</u>	<u>63</u>
3-3	<u>Using Edition-Based Redefinition: Sample Code</u>	<u>65</u>
3-4	<u>Using Transaction Guard: Sample Code</u>	<u>83</u>
3-5	<u>Using the add Element with bindinfo</u>	<u>146</u>
3-6	<u>Using the add Element with metadata</u>	<u>146</u>
6-1	<u>Using OracleAccessToken Constructor and Event</u>	<u>3</u>

List of Tables

1-1	<u>Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client</u>	<u>5</u>
1-2	<u>Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Enumerations</u>	<u>12</u>
1-3	<u>Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Structures</u>	<u>15</u>
1-4	<u>Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Exceptions</u>	<u>15</u>
1-5	<u>Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Classes</u>	<u>16</u>
1-6	<u>Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Interfaces</u>	<u>17</u>
1-7	<u>Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Enumerations</u>	<u>18</u>
1-8	<u>Application Programming Interfaces not supported in ODP.NET, Managed Driver and ODP.NET Core</u>	<u>18</u>
2-1	<u>ODP.NET, Managed Driver Files with Descriptions</u>	<u>8</u>
2-2	<u>ODP.NET, Core Files with Descriptions</u>	<u>10</u>
2-3	<u>Configuration Attributes</u>	<u>18</u>
2-4	<u>Encryption Algorithms for ODP.NET, Managed Driver</u>	<u>39</u>
2-5	<u>Microsoft Active Directory: Encryption Types and Authentication Credentials For Connecting and Binding</u>	<u>41</u>
2-6	<u>Oracle Internet Directory: Encryption Types and Authentication Credentials For Connecting and Binding</u>	<u>41</u>
2-7	<u>Required ONS Configuration Parameters</u>	<u>45</u>
2-8	<u>Optional ONS Configuration Parameters</u>	<u>46</u>
2-9	<u>Configuration Differences among ODP.NET Core, ODP.NET, Unmanaged Driver and ODP.NET, Managed Driver</u>	<u>51</u>
2-10	<u>Azure Authentication Methods and their Values</u>	<u>53</u>
2-11	<u>OCI Authentication Methods and their Values</u>	<u>55</u>
3-1	<u>ODP.NET Classes that Inherit from ADO.NET Base Classes</u>	<u>2</u>
3-2	<u>ODP.NET Authentication using IAM and Signature Token</u>	<u>13</u>
3-3	<u>ODP.NET OCI IAM Parameters</u>	<u>18</u>
3-4	<u>ODP.NET OCI IAM Authentication Flows</u>	<u>19</u>
3-5	<u>ODP.NET OCI IAM Configuration Locations</u>	<u>19</u>
3-6	<u>Supported Connection String Attributes</u>	<u>28</u>
3-7	<u>Performance Counters for Connection Pooling</u>	<u>57</u>
3-8	<u>Field Names of Performance Counters and Maximum Number of Characters</u>	<u>59</u>
3-9	<u>Configurations for ODP.NET Driver Types</u>	<u>75</u>
3-10	<u>Implication of Committed and UserCallCompleted Values</u>	<u>83</u>
3-11	<u>ODP.NET TAF Failover Mode Support</u>	<u>87</u>
3-12	<u>ODP.NET TAF Failover Method Support</u>	<u>88</u>
3-13	<u>Supported Operations for Shard Key Types</u>	<u>92</u>

3-14	<u>ODP.NET Distributed Transaction Support Matrix</u>	98
3-15	<u>OracleDbType Enumeration Values</u>	103
3-16	<u>Inference of System.Data.DbType from OracleDbType</u>	104
3-17	<u>Inference of OracleDbType from DbType</u>	105
3-18	<u>Inference of DbType and OracleDbType from Value (.NET Datatypes)</u>	106
3-19	<u>Inference of DbType and OracleDbType from Value (ODP.NET Types)</u>	107
3-20	<u>OracleParameterStatus Members</u>	114
3-21	<u>Value Property Type of ODP.NET Type</u>	120
3-22	<u>.NET Type Accessors</u>	127
3-23	<u>ODP.NET Type Accessors</u>	130
3-24	<u>Supported OracleDataReader CLOB Methods for InitialLOBFetchSize of -1 and LegacyEntireLobFetch of 1</u>	135
3-25	<u>Supported OracleDataReader BLOB Methods for InitialLOBFetchSize of -1 and LegacyEntireLobFetch of 1</u>	135
3-26	<u>Allowed Parameters in Attributes List</u>	144
3-27	<u>ODP.NET LOB Objects</u>	151
3-28	<u>Dense Vector Numeric Formats</u>	157
3-29	<u>Sparse Vector Numeric Formats</u>	157
3-30	<u>Characters with Special Meaning in XML</u>	169
3-31	<u>Attribute Mappings Between UDTs and Custom Object Types</u>	185
3-32	<u>Type and Value Returned from OracleDataReader Object</u>	187
3-33	<u>Values Returned from OracleDataReader Methods</u>	188
3-34	<u>Valid Ways to Bind Input Parameters for Oracle UDTs</u>	189
3-35	<u>Valid Ways to Bind Output Parameters for Oracle UDTs</u>	190
3-36	<u>Types that Populate the DataSet with ADO.NET 2.0</u>	191
3-37	<u>OracleRef class members</u>	195
3-38	<u>Mapping AQ Features with their ODP.NET Implementation</u>	206
3-39	<u>Oracle NUMBER to .NET Decimal Comparisons</u>	217
3-40	<u>Oracle Date to .NET DateTime Comparisons</u>	217
3-41	<u>Oracle TimeStamp to .NET DateTime Comparisons</u>	218
3-42	<u>Oracle INTERVAL DAY TO SECOND to .NET TimeSpan Comparisons</u>	218
3-43	<u>ODP.NET OpenTelemetry activity tags</u>	229
3-44	<u>Metric Counters and Levels</u>	236
4-1	<u>Mapping of Oracle Data Types and EDM Types</u>	3
4-2	<u>EDM Type Facets for Bfile</u>	6
4-3	<u>EDM Type Facets for Blob</u>	6
4-4	<u>EDM Type Facets for Char</u>	7

4-5	<u>EDM Type Facets for Clob</u>	<u>7</u>
4-6	<u>EDM Type Facets for Date</u>	<u>7</u>
4-7	<u>EDM Type Facets for Float</u>	<u>7</u>
4-8	<u>EDM Type Facets for Interval Day To Second</u>	<u>8</u>
4-9	<u>EDM Type Facets for Interval Year To Month</u>	<u>8</u>
4-10	<u>EDM Type Facets for Json</u>	<u>9</u>
4-11	<u>EDM Type Facets for Long</u>	<u>9</u>
4-12	<u>EDM Type Facets for Long Raw</u>	<u>9</u>
4-13	<u>EDM Type Facets for NChar</u>	<u>9</u>
4-14	<u>EDM Type Facets for NClob</u>	<u>10</u>
4-15	<u>EDM Type Facets for Number</u>	<u>10</u>
4-16	<u>EDM Type Facets for NVarchar2</u>	<u>11</u>
4-17	<u>EDM Type Facets for Raw</u>	<u>11</u>
4-18	<u>EDM Type Facets for ROWID</u>	<u>11</u>
4-19	<u>EDM Type Facets for Timestamp</u>	<u>12</u>
4-20	<u>EDM Type Facets for Timestamp with Local Time Zone</u>	<u>12</u>
4-21	<u>EDM Type Facets for Timestamp with Time Zone</u>	<u>12</u>
4-22	<u>EDM Type Facets for UROWID</u>	<u>12</u>
4-23	<u>EDM Type Facets for Varchar2</u>	<u>13</u>
4-24	<u>EDM Type Facets for XMLType</u>	<u>13</u>
4-25	<u>Mapping of .NET Data Types to Oracle Data Types</u>	<u>18</u>
4-26	<u>Mapping of Data Annotations and the Code First Fluent APIs</u>	<u>20</u>
5-1	<u>ODP.NET Entity Framework Core Migrations Data Type Default Mappings</u>	<u>24</u>
5-2	<u>ODP.NET Entity Framework Core Reverse Engineering Data Type Default Mappings</u>	<u>27</u>
5-3	<u>Schema and Table Filter Mapping</u>	<u>28</u>
5-4	<u>ODP.NET Entity Framework Core Scaffolding Default and Alternative Data Type Mappings</u>	<u>29</u>
5-5	<u>Oracle Vector EF Core Data Type Mapping</u>	<u>33</u>
5-6	<u>NetTopologySuite Methods and Oracle Spatial Functions Mapping</u>	<u>41</u>
5-7	<u>NetTopologySuite Properties and Oracle Spatial Functions Mapping</u>	<u>42</u>
6-1	<u>OracleAccessToken Constructor</u>	<u>3</u>
6-2	<u>OracleAccessToken Event</u>	<u>4</u>
6-3	<u>OracleClientFactory Field</u>	<u>8</u>
6-4	<u>OracleClientFactory Constructor</u>	<u>8</u>
6-5	<u>OracleClientFactory Public Properties</u>	<u>8</u>
6-6	<u>OracleClientFactory Public Method</u>	<u>8</u>
6-7	<u>OracleClientFactory Field</u>	<u>9</u>
6-8	<u>OracleClientFactory Public Properties</u>	<u>10</u>

6-9	<u>OracleClientFactory Public Method</u>	<u>11</u>
6-10	<u>OracleCommand Constructors</u>	<u>18</u>
6-11	<u>OracleCommand Static Method</u>	<u>18</u>
6-12	<u>OracleCommand Properties</u>	<u>19</u>
6-13	<u>OracleCommand Public Methods</u>	<u>20</u>
6-14	<u>OracleCommand Static Method</u>	<u>23</u>
6-15	<u>OracleCommand Properties</u>	<u>24</u>
6-16	<u>OracleCommand Public Methods</u>	<u>46</u>
6-17	<u>OracleCommandBuilder Constructors</u>	<u>75</u>
6-18	<u>OracleCommandBuilder Static Methods</u>	<u>75</u>
6-19	<u>OracleCommandBuilder Properties</u>	<u>75</u>
6-20	<u>OracleCommandBuilder Public Methods</u>	<u>76</u>
6-21	<u>OracleCommandBuilder Events</u>	<u>76</u>
6-22	<u>OracleCommandBuilder Static Methods</u>	<u>78</u>
6-23	<u>OracleCommandBuilder Properties</u>	<u>81</u>
6-24	<u>OracleCommandBuilder Public Methods</u>	<u>87</u>
6-25	<u>OracleCommandBuilder Event</u>	<u>96</u>
6-26	<u>OracleConfiguration Methods</u>	<u>99</u>
6-27	<u>OracleConfiguration Connection Properties</u>	<u>99</u>
6-28	<u>OracleConfiguration Secure Connection Properties</u>	<u>100</u>
6-29	<u>OracleConfiguration Command Properties</u>	<u>102</u>
6-30	<u>OracleConfiguration Directories Properties</u>	<u>102</u>
6-31	<u>OracleConfiguration High Availability Properties</u>	<u>103</u>
6-32	<u>OracleConfiguration Performance Properties</u>	<u>103</u>
6-33	<u>OracleConfiguration Diagnostics and Tracing Properties</u>	<u>104</u>
6-34	<u>OracleConfiguration Methods</u>	<u>105</u>
6-35	<u>OracleConfiguration Connection Properties</u>	<u>108</u>
6-36	<u>ConfigurationProviders Values</u>	<u>110</u>
6-37	<u>OracleConfiguration Secure Connection Properties</u>	<u>124</u>
6-38	<u>OracleTokenAuth Members</u>	<u>138</u>
6-39	<u>OracleConfiguration Command Properties</u>	<u>142</u>
6-40	<u>OracleConfiguration Directories Properties</u>	<u>147</u>
6-41	<u>OracleConfiguration High Availability Properties</u>	<u>152</u>
6-42	<u>OracleConfiguration Performance Properties</u>	<u>161</u>
6-43	<u>OracleConfiguration Diagnostics and Tracing Properties</u>	<u>168</u>
6-44	<u>OracleConnection Constructors</u>	<u>177</u>
6-45	<u>OracleConnection Static Methods</u>	<u>177</u>

6-46	<u>OracleConnection Properties</u>	<u>178</u>
6-47	<u>OracleConnection Public Methods</u>	<u>183</u>
6-48	<u>OracleConnection Events</u>	<u>185</u>
6-49	<u>OracleConnection Static Methods</u>	<u>188</u>
6-50	<u>OracleConnection Properties</u>	<u>193</u>
6-51	<u>Supported Connection String Attributes</u>	<u>209</u>
6-52	<u>OracleTokenAuth Members</u>	<u>258</u>
6-53	<u>OracleConnection Public Methods</u>	<u>266</u>
6-54	<u>OracleConnection Events</u>	<u>313</u>
6-55	<u>OracleConnectionOpenEventArgs Properties</u>	<u>319</u>
6-56	<u>OracleConnectionOpenEventArgs Properties</u>	<u>320</u>
6-57	<u>OracleConnectionStringBuilder Constructors</u>	<u>325</u>
6-58	<u>OracleConnectionStringBuilder Public Properties</u>	<u>325</u>
6-59	<u>OracleConnectionStringBuilder Public Methods</u>	<u>327</u>
6-60	<u>OracleConnectionStringBuilder Public Properties</u>	<u>329</u>
6-61	<u>OracleConnectionStringBuilder Public Methods</u>	<u>348</u>
6-62	<u>OracleCredential Constructors</u>	<u>354</u>
6-63	<u>OracleCredential Properties</u>	<u>354</u>
6-64	<u>OracleCredential Public Methods</u>	<u>354</u>
6-65	<u>OracleCredential Properties</u>	<u>357</u>
6-66	<u>OracleCredential Public Methods</u>	<u>360</u>
6-67	<u>OracleDataAdapter Constructors</u>	<u>364</u>
6-68	<u>OracleDataAdapter Static Method</u>	<u>364</u>
6-69	<u>OracleDataAdapter Properties</u>	<u>364</u>
6-70	<u>OracleDataAdapter Public Methods</u>	<u>365</u>
6-71	<u>OracleDataAdapter Events</u>	<u>366</u>
6-72	<u>OracleDataAdapter Static Method</u>	<u>369</u>
6-73	<u>OracleDataAdapter Properties</u>	<u>370</u>
6-74	<u>OracleDataAdapter Public Methods</u>	<u>378</u>
6-75	<u>OracleDataAdapter Events</u>	<u>384</u>
6-76	<u>OracleDatabase Constructors</u>	<u>389</u>
6-77	<u>OracleDatabase Properties</u>	<u>389</u>
6-78	<u>OracleDatabase Public Methods</u>	<u>389</u>
6-79	<u>OracleDatabase Properties</u>	<u>392</u>
6-80	<u>OracleDatabase Public Methods</u>	<u>392</u>
6-81	<u>OracleDataReader Static Method</u>	<u>403</u>
6-82	<u>OracleDataReader Properties</u>	<u>403</u>

6-83	<u>OracleDataReader Public Methods</u>	404
6-84	<u>OracleDataReader Static Method</u>	408
6-85	<u>OracleDataReader Properties</u>	408
6-86	<u>OracleDataReader Public Methods</u>	422
6-87	<u>OracleDataReader SchemaTable</u>	488
6-88	<u>OracleDataSource Public Methods</u>	511
6-89	<u>OracleDataSource Properties</u>	511
6-90	<u>OracleDataSource Public Events</u>	512
6-91	<u>OracleDataSource Public Methods</u>	512
6-92	<u>OracleDataSource Properties</u>	516
6-93	<u>OracleDataSource Public Events</u>	520
6-94	<u>OracleDataSourceBuilder Constructors</u>	522
6-95	<u>OracleDataSourceBuilder Public Methods</u>	522
6-96	<u>OracleDataSourceBuilder Properties</u>	522
6-97	<u>OracleDataSourceBuilder Public Methods</u>	523
6-98	<u>OracleDataSourceBuilder Properties</u>	527
6-99	<u>OracleDataSourceCollection Methods</u>	535
6-100	<u>OracleDataSourceCollection Properties</u>	535
6-101	<u>OracleDataSourceCollection Methods</u>	535
6-102	<u>OracleDataSourceCollection Properties</u>	537
6-103	<u>OracleDataSourceCommand Public Methods</u>	539
6-104	<u>OracleDataSourceCommand Properties</u>	539
6-105	<u>OracleDataSourceCommand Public Methods</u>	540
6-106	<u>OracleDataSourceCommand Properties</u>	542
6-107	<u>OracleDataSourceEnumerator Method</u>	544
6-108	<u>OracleDataSourceEnumerator Method</u>	545
6-109	<u>OracleDataSourceEnumerator Method</u>	545
6-110	<u>OracleDBMSAssert Methods</u>	548
6-111	<u>OracleError Static Method</u>	560
6-112	<u>OracleError Properties</u>	560
6-113	<u>OracleError Methods</u>	560
6-114	<u>OracleError Static Method</u>	561
6-115	<u>OracleError Properties</u>	561
6-116	<u>OracleError Methods</u>	565
6-117	<u>OracleErrorCollection Static Methods</u>	567
6-118	<u>OracleErrorCollection Properties</u>	567
6-119	<u>OracleErrorCollection Public Methods</u>	568

6-120	<u>OracleErrorCollection Static Method</u>	568
6-121	<u>OracleErrorCollection Properties</u>	568
6-122	<u>OracleErrorCollection Public Methods</u>	569
6-123	<u>OracleException Static Method</u>	571
6-124	<u>OracleException Properties</u>	571
6-125	<u>OracleException Methods</u>	572
6-126	<u>OracleException Static Method</u>	573
6-127	<u>OracleException Properties</u>	573
6-128	<u>OracleException Methods</u>	578
6-129	<u>OracleInfoMessageEventArgs Static Method</u>	582
6-130	<u>OracleInfoMessageEventArgs Properties</u>	582
6-131	<u>OracleInfoMessageEventArgs Public Methods</u>	583
6-132	<u>OracleInfoMessageEventArgs Static Method</u>	583
6-133	<u>OracleInfoMessageEventArgs Properties</u>	583
6-134	<u>OracleInfoMessageEventArgs Public Methods</u>	585
6-135	<u>OracleLogicalTransaction Public Read-Only Properties</u>	588
6-136	<u>OracleLogicalTransaction Methods</u>	588
6-137	<u>OracleLogicalTransaction Public Read-Only Properties</u>	589
6-138	<u>Outcome of OracleLogicalTransaction Committed and UserCallCompleted Properties</u>	590
6-139	<u>OracleLogicalTransaction Methods</u>	592
6-140	<u>OracleOnsServerCollection Methods</u>	596
6-141	<u>OracleOnsServerCollection Properties</u>	596
6-142	<u>OracleOnsServerCollection Methods</u>	596
6-143	<u>OracleOnsServerCollection Properties</u>	598
6-144	<u>OracleOpaqueString Constructors</u>	600
6-145	<u>OracleOpaqueString Methods</u>	600
6-146	<u>OracleOpaqueString Properties</u>	601
6-147	<u>OracleOpaqueString Properties</u>	601
6-148	<u>OracleOpaqueString Methods</u>	602
6-149	<u>OracleParameter Constructors</u>	607
6-150	<u>OracleParameter Static Methods</u>	607
6-151	<u>OracleParameter Properties</u>	608
6-152	<u>OracleParameter Public Methods</u>	609
6-153	<u>OracleParameter Static Method</u>	620
6-154	<u>OracleParameter Properties</u>	621
6-155	<u>OracleParameter Public Methods</u>	638
6-156	<u>OracleParameterCollection Static Methods</u>	643

6-157	<u>OracleParameterCollection Properties</u>	644
6-158	<u>OracleParameterCollection Public Methods</u>	644
6-159	<u>OracleParameterCollection Static Method</u>	645
6-160	<u>OracleParameterCollection Properties</u>	645
6-161	<u>OracleParameterCollection Public Methods</u>	650
6-162	<u>OraclePermission Constructor</u>	671
6-163	<u>OraclePermission Static Methods</u>	672
6-164	<u>OraclePermission Public Properties</u>	672
6-165	<u>OraclePermission Public Methods</u>	672
6-166	<u>OraclePermission Static Methods</u>	673
6-167	<u>OraclePermission Public Properties</u>	674
6-168	<u>OraclePermission Public Methods</u>	674
6-169	<u>OraclePermission Constructor</u>	679
6-170	<u>OraclePermissionAttribute Static Methods</u>	679
6-171	<u>OraclePermissionAttribute Public Properties</u>	679
6-172	<u>OraclePermissionAttribute Public Methods</u>	680
6-173	<u>OraclePermissionAttribute Static Methods</u>	681
6-174	<u>OraclePermissionAttribute Public Properties</u>	681
6-175	<u>OraclePermissionAttribute Public Methods</u>	682
6-176	<u>OracleRefreshAccessTokenEventArgs Properties</u>	684
6-177	<u>OracleRefreshAccessTokenEventArgs Properties</u>	684
6-178	<u>OracleRoleCollection Methods</u>	687
6-179	<u>OracleRoleCollection Properties</u>	687
6-180	<u>OracleRoleCollection Methods</u>	688
6-181	<u>OracleRoleCollection Properties</u>	693
6-182	<u>OracleRowUpdatedEventArgs Constructors</u>	697
6-183	<u>OracleRowUpdatedEventArgs Static Method</u>	697
6-184	<u>OracleRowUpdatedEventArgs Properties</u>	697
6-185	<u>OracleRowUpdatedEventArgs Public Methods</u>	697
6-186	<u>OracleRowUpdatedEventArgs Static Method</u>	698
6-187	<u>OracleRowUpdatedEventArgs Properties</u>	699
6-188	<u>OracleRowUpdatedEventArgs Public Methods</u>	700
6-189	<u>OracleRowUpdatingEventArgs Constructors</u>	702
6-190	<u>OracleRowUpdatingEventArgs Static Methods</u>	703
6-191	<u>OracleRowUpdatingEventArgs Properties</u>	703
6-192	<u>OracleRowUpdatingEventArgs Public Methods</u>	703
6-193	<u>OracleRowUpdatingEventArgs Static Method</u>	704

6-194	<u>OracleRowUpdatingEventArgs Properties</u>	705
6-195	<u>OracleRowUpdatingEventArgs Public Methods</u>	706
6-196	<u>OracleSessionlessTransactionOptions Constructors</u>	708
6-197	<u>OracleSessionlessTransactionOptions Properties</u>	708
6-198	<u>OracleSessionlessTransactionOptions Properties</u>	711
6-199	<u>OracleShardingKey Constructors</u>	715
6-200	<u>OracleShardingKey Instance Methods</u>	715
6-201	<u>OracleShardingKey Instance Methods</u>	717
6-202	<u>OracleSparseVector Constructors</u>	719
6-203	<u>OracleSparseVector Properties</u>	720
6-204	<u>OracleSparseVector Static Field</u>	720
6-205	<u>OracleSparseVector Instance Methods</u>	720
6-206	<u>OracleSparseVector Properties</u>	726
6-207	<u>OracleSparseVector Instance Methods</u>	731
6-208	<u>OracleTransaction Static Method</u>	739
6-209	<u>OracleTransaction Properties</u>	739
6-210	<u>OracleTransaction Public Methods</u>	739
6-211	<u>OracleTransaction Static Method</u>	740
6-212	<u>OracleTransaction Properties</u>	741
6-213	<u>OracleTransaction Public Methods</u>	742
6-214	<u>OracleVector Constructors</u>	751
6-215	<u>OracleVector Properties</u>	751
6-216	<u>OracleVector Static Field</u>	752
6-217	<u>OracleVector Instance Methods</u>	752
6-218	<u>OracleVector Properties</u>	757
6-219	<u>OracleVector Instance Methods</u>	761
6-220	<u>OracleAllowedLogonVersionClient Enumeration Values</u>	765
6-221	<u>OracleCollectionType Enumeration Values</u>	766
6-222	<u>OracleConnectionOpenReason Enumeration Values</u>	767
6-223	<u>OracleConnectionType Enumeration Values</u>	767
6-224	<u>OracleDBAPrivilege Enumeration Values</u>	768
6-225	<u>OracleDBShutdownMode Enumeration Values</u>	769
6-226	<u>OracleDBStartupMode Enumeration Values</u>	770
6-227	<u>OracleDbType Enumeration Values</u>	771
6-228	<u>OracleDRCPPurity Enumeration Values</u>	773
6-229	<u>OracleIdentityType Members</u>	773
6-230	<u>OracleParameterStatus Members</u>	774

6-231	<u>OraclePasswordAuth Members</u>	<u>775</u>
6-232	<u>OracleSessionlessTransactionStartBehavior Members</u>	<u>776</u>
6-233	<u>OracleTokenAuth Members</u>	<u>777</u>
6-234	<u>OracleVectorType Members</u>	<u>778</u>
7-1	<u>OracleXmlCommandType Members</u>	<u>1</u>
7-2	<u>OracleXmlQueryProperties Constructors</u>	<u>5</u>
7-3	<u>OracleXmlQueryProperties Properties</u>	<u>6</u>
7-4	<u>OracleXmlQueryProperties Public Methods</u>	<u>6</u>
7-5	<u>OracleXmlQueryProperties Properties</u>	<u>7</u>
7-6	<u>OracleXmlQueryProperties Public Methods</u>	<u>10</u>
7-7	<u>OracleXmlSaveProperties Constructor</u>	<u>14</u>
7-8	<u>OracleXmlSaveProperties Properties</u>	<u>14</u>
7-9	<u>OracleXmlSaveProperties Public Methods</u>	<u>15</u>
7-10	<u>OracleXmlSaveProperties Properties</u>	<u>15</u>
7-11	<u>OracleXmlSaveProperties Public Methods</u>	<u>19</u>
7-12	<u>OracleXmlStream Constructors</u>	<u>21</u>
7-13	<u>OracleXmlStream Static Methods</u>	<u>21</u>
7-14	<u>OracleXmlStream Instance Properties</u>	<u>21</u>
7-15	<u>OracleXmlStream Instance Methods</u>	<u>22</u>
7-16	<u>OracleXmlStream Static Methods</u>	<u>24</u>
7-17	<u>OracleXmlStream Instance Properties</u>	<u>24</u>
7-18	<u>OracleXmlStream Instance Methods</u>	<u>28</u>
7-19	<u>OracleXmlType Constructors</u>	<u>41</u>
7-20	<u>OracleXmlType Static Methods</u>	<u>41</u>
7-21	<u>OracleXmlType Static Field</u>	<u>41</u>
7-22	<u>OracleXmlType Instance Properties</u>	<u>41</u>
7-23	<u>OracleXmlType Instance Methods</u>	<u>42</u>
7-24	<u>OracleXmlType Static Methods</u>	<u>46</u>
7-25	<u>OracleXmlType Static Field</u>	<u>46</u>
7-26	<u>OracleXmlType Instance Properties</u>	<u>47</u>
7-27	<u>OracleXmlType Instance Methods</u>	<u>53</u>
8-1	<u>OracleHAEventArgs Properties</u>	<u>2</u>
8-2	<u>OracleHAEventArgs Properties</u>	<u>3</u>
8-3	<u>OracleHAEventSource Enumeration Member Values</u>	<u>9</u>
8-4	<u>OracleHAEventStatus Enumeration Values</u>	<u>10</u>
9-1	<u>OracleDependency Constructors</u>	<u>2</u>
9-2	<u>OracleDependency Static Fields</u>	<u>3</u>

9-3	<u>OracleDependency Static Methods</u>	<u>3</u>
9-4	<u>OracleDependency Properties</u>	<u>3</u>
9-5	<u>OracleDependency Methods</u>	<u>4</u>
9-6	<u>OracleDependency Events</u>	<u>4</u>
9-7	<u>OracleDependency Static Fields</u>	<u>8</u>
9-8	<u>OracleDependency Static Methods</u>	<u>10</u>
9-9	<u>OracleDependency Properties</u>	<u>11</u>
9-10	<u>OracleDependency Methods</u>	<u>17</u>
9-11	<u>OracleDependency Event</u>	<u>20</u>
9-12	<u>OracleNotificationRequest Static Method</u>	<u>22</u>
9-13	<u>OracleNotificationRequest Properties</u>	<u>22</u>
9-14	<u>OracleNotificationRequest Methods</u>	<u>22</u>
9-15	<u>OracleNotificationRequest Static Method</u>	<u>23</u>
9-16	<u>OracleNotificationRequest Properties</u>	<u>23</u>
9-17	<u>OracleNotificationRequest Methods</u>	<u>27</u>
9-18	<u>OracleNotificationEventArgs Static Field</u>	<u>29</u>
9-19	<u>OracleNotificationEventArgs Static Method</u>	<u>29</u>
9-20	<u>OracleNotificationEventArgs Properties</u>	<u>29</u>
9-21	<u>OracleNotificationEventArgs Methods</u>	<u>30</u>
9-22	<u>OracleNotificationEventArgs Static Field</u>	<u>30</u>
9-23	<u>OracleNotificationEventArgs Static Method</u>	<u>30</u>
9-24	<u>OracleNotificationEventArgs Properties</u>	<u>31</u>
9-25	<u>DataTable Object Column Data</u>	<u>32</u>
9-26	<u>OracleNotificationEventArgs Methods</u>	<u>35</u>
9-27	<u>OracleRowidInfo Members</u>	<u>37</u>
9-28	<u>OracleNotificationType Members</u>	<u>38</u>
9-29	<u>OracleNotificationSource Members</u>	<u>38</u>
9-30	<u>OracleNotificationInfo Members</u>	<u>39</u>
10-1	<u>OracleGlobalization Static Methods</u>	<u>3</u>
10-2	<u>OracleGlobalization Properties</u>	<u>3</u>
10-3	<u>OracleGlobalization Public Methods</u>	<u>4</u>
10-4	<u>OracleGlobalization Static Methods</u>	<u>4</u>
10-5	<u>OracleGlobalization Properties</u>	<u>10</u>
10-6	<u>OracleGlobalization Public Methods</u>	<u>21</u>
10-7	<u>OracleDatabaseCharset Enumeration Member Name and Description</u>	<u>23</u>
10-8	<u>OracleDatabaseNCharset Enumeration</u>	<u>29</u>
11-1	<u>OracleFailoverEventArgs Static Methods</u>	<u>3</u>

11-2	<u>OracleFailoverEventArgs Properties</u>	<u>4</u>
11-3	<u>OracleFailoverEventArgs Public Methods</u>	<u>4</u>
11-4	<u>OracleFailoverEventArgs Static Methods</u>	<u>4</u>
11-5	<u>OracleFailoverEventArgs Properties</u>	<u>5</u>
11-6	<u>OracleFailoverEventArgs Public Methods</u>	<u>7</u>
11-7	<u>FailoverEvent Enumeration Values</u>	<u>11</u>
11-8	<u>FailoverReturnCode Enumeration Values</u>	<u>12</u>
11-9	<u>FailoverType Enumeration Values</u>	<u>12</u>
11-10	<u>OracleFailoverEvent Enumeration Values</u>	<u>13</u>
11-11	<u>OracleFailoverRestore Enumeration Values</u>	<u>15</u>
11-12	<u>OracleFailoverReturnCode Enumeration Values</u>	<u>16</u>
11-13	<u>OracleFailoverType Enumeration Values</u>	<u>17</u>
12-1	<u>OracleAQAgent Constructors</u>	<u>2</u>
12-2	<u>OracleAQAgent Properties</u>	<u>3</u>
12-3	<u>OracleAQAgent Properties</u>	<u>5</u>
12-4	<u>OracleAQDequeueOptions Constructor</u>	<u>7</u>
12-5	<u>OracleAQDequeueOptions Properties</u>	<u>8</u>
12-6	<u>OracleAQDequeueOptions Public Methods</u>	<u>8</u>
12-7	<u>OracleAQDequeueOptions Properties</u>	<u>9</u>
12-8	<u>OracleAQDequeueOptions Public Methods</u>	<u>15</u>
12-9	<u>OracleAQEnqueueOptions Constructor</u>	<u>17</u>
12-10	<u>OracleAQEnqueueOptions Properties</u>	<u>17</u>
12-11	<u>OracleAQEnqueueOptions Public Methods</u>	<u>17</u>
12-12	<u>OracleAQEnqueueOptions Properties</u>	<u>18</u>
12-13	<u>OracleAQEnqueueOptions Public Methods</u>	<u>20</u>
12-14	<u>OracleAQMessage Constructors</u>	<u>22</u>
12-15	<u>OracleAQMessage Properties</u>	<u>22</u>
12-16	<u>OracleAQMessage Properties</u>	<u>24</u>
12-17	<u>OracleAQMessageAvailableEventArgs Constructor</u>	<u>35</u>
12-18	<u>OracleAQMessageAvailableEventArgs Properties</u>	<u>35</u>
12-19	<u>OracleAQMessageAvailableEventArgs Properties</u>	<u>36</u>
12-20	<u>OracleAQQueue Constructors</u>	<u>47</u>
12-21	<u>OracleAQQueue Static Methods</u>	<u>47</u>
12-22	<u>OracleAQQueue Properties</u>	<u>47</u>
12-23	<u>OracleAQQueue Public Methods</u>	<u>48</u>
12-24	<u>OracleAQQueue Events</u>	<u>48</u>
12-25	<u>OracleAQQueue Properties</u>	<u>49</u>

12-26	<u>Message Types and Payloads</u>	52
12-27	<u>Payload Types for Dequeued Messages</u>	52
12-28	<u>OracleAQQueue Static Methods</u>	59
12-29	<u>OracleAQQueue Public Methods</u>	62
12-30	<u>OracleAQQueue Events</u>	81
12-31	<u>OracleAQDequeueMode Members</u>	86
12-32	<u>OracleAQMessageDeliveryMode Members</u>	87
12-33	<u>OracleAQMessageState Members</u>	88
12-34	<u>OracleAQMessageType Members</u>	89
12-35	<u>OracleAQNavigationMode Members</u>	90
12-36	<u>OracleAQNotificationGroupingType Members</u>	91
12-37	<u>OracleAQNotificationType Members</u>	91
12-38	<u>OracleAQVisibilityMode Members</u>	92
12-39	<u>OracleTxEventQNotificationDirective Members</u>	93
13-1	<u>OracleBFile Constructors</u>	3
13-2	<u>OracleBFile Static Fields</u>	4
13-3	<u>OracleBFile Static Methods</u>	4
13-4	<u>OracleBFile Instance Properties</u>	4
13-5	<u>OracleBFile Instance Methods</u>	5
13-6	<u>OracleBFile Static Fields</u>	8
13-7	<u>OracleBFile Static Methods</u>	9
13-8	<u>OracleBFile Instance Properties</u>	9
13-9	<u>OracleBFile Instance Methods</u>	17
13-10	<u>OracleBlob Constructors</u>	55
13-11	<u>OracleBlob Static Fields</u>	55
13-12	<u>OracleBlob Static Methods</u>	56
13-13	<u>OracleBlob Instance Properties</u>	56
13-14	<u>OracleBlob Instance Methods</u>	56
13-15	<u>OracleBlob Static Fields</u>	60
13-16	<u>OracleBlob Static Methods</u>	61
13-17	<u>OracleBlob Instance Properties</u>	61
13-18	<u>OracleBlob Instance Methods</u>	68
13-19	<u>OracleClob Constructors</u>	105
13-20	<u>OracleClob Static Fields</u>	105
13-21	<u>OracleClob Static Methods</u>	105
13-22	<u>OracleClob Instance Properties</u>	106
13-23	<u>OracleClob Instance Methods</u>	106

13-24	<u>OracleClob Static Fields</u>	<u>110</u>
13-25	<u>OracleClob Static Methods</u>	<u>111</u>
13-26	<u>OracleClob Instance Properties</u>	<u>111</u>
13-27	<u>OracleClob Instance Methods</u>	<u>119</u>
13-28	<u>OracleRefCursor Static Methods</u>	<u>166</u>
13-29	<u>OracleRefCursor Static Field</u>	<u>166</u>
13-30	<u>OracleRefCursor Properties</u>	<u>166</u>
13-31	<u>OracleRefCursor Instance Methods</u>	<u>167</u>
13-32	<u>OracleRefCursor Static Methods</u>	<u>167</u>
13-33	<u>OracleRefCursor Static Field</u>	<u>168</u>
13-34	<u>OracleRefCursor Properties</u>	<u>169</u>
13-35	<u>OracleRefCursor Instance Methods</u>	<u>173</u>
14-1	<u>OracleBinary Constructors</u>	<u>3</u>
14-2	<u>OracleBinary Static Fields</u>	<u>3</u>
14-3	<u>OracleBinary Static Methods</u>	<u>3</u>
14-4	<u>OracleBinary Static Operators</u>	<u>4</u>
14-5	<u>OracleBinary Static Type Conversion Operators</u>	<u>4</u>
14-6	<u>OracleBinary Properties</u>	<u>4</u>
14-7	<u>OracleBinary Instance Methods</u>	<u>4</u>
14-8	<u>OracleBinary Static Fields</u>	<u>5</u>
14-9	<u>OracleBinary Static Methods</u>	<u>6</u>
14-10	<u>OracleBinary Static Operators</u>	<u>13</u>
14-11	<u>OracleBinary Static Type Conversion Operators</u>	<u>19</u>
14-12	<u>OracleBinary Properties</u>	<u>20</u>
14-13	<u>OracleBinary Instance Methods</u>	<u>23</u>
14-14	<u>Accepted OracleBoolean Values</u>	<u>27</u>
14-15	<u>OracleBoolean Constructors</u>	<u>28</u>
14-16	<u>OracleBoolean Static Fields</u>	<u>29</u>
14-17	<u>OracleBoolean Static Methods</u>	<u>29</u>
14-18	<u>OracleBoolean Static Operators</u>	<u>30</u>
14-19	<u>OracleBoolean Static Type Conversions</u>	<u>30</u>
14-20	<u>OracleBoolean Properties</u>	<u>30</u>
14-21	<u>OracleBoolean Instance Methods</u>	<u>31</u>
14-22	<u>OracleBoolean Static Fields</u>	<u>33</u>
14-23	<u>OracleBoolean Static Methods</u>	<u>35</u>
14-24	<u>OracleBoolean Static Operators</u>	<u>43</u>
14-25	<u>OracleBoolean Static Type Conversions</u>	<u>52</u>

14-26	<u>OracleBoolean Properties</u>	59
14-27	<u>OracleBoolean Instance Methods</u>	61
14-28	<u>OracleDate Constructors</u>	66
14-29	<u>OracleDate Static Fields</u>	66
14-30	<u>OracleDate Static Methods</u>	67
14-31	<u>OracleDate Static Operators</u>	67
14-32	<u>OracleDate Static Type Conversions</u>	68
14-33	<u>OracleDate Properties</u>	68
14-34	<u>OracleDate Methods</u>	68
14-35	<u>OracleDate Static Fields</u>	73
14-36	<u>OracleDate Static Methods</u>	75
14-37	<u>OracleDate Static Operators</u>	82
14-38	<u>OracleDate Static Type Conversions</u>	87
14-39	<u>OracleDate Properties</u>	91
14-40	<u>OracleDate Methods</u>	96
14-41	<u>OracleDecimal Constructors</u>	103
14-42	<u>OracleDecimal Static Fields</u>	103
14-43	<u>OracleDecimal Static (Comparison) Methods</u>	103
14-44	<u>OracleDecimal Static (Manipulation) Methods</u>	104
14-45	<u>OracleDecimal Static (Logarithmic) Methods</u>	105
14-46	<u>OracleDecimal Static (Trigonometric) Methods</u>	105
14-47	<u>OracleDecimal Static (Comparison) Operators</u>	105
14-48	<u>OracleDecimal Static Operators (Conversion from .NET Type to OracleDecimal)</u>	106
14-49	<u>OracleDecimal Static Operators (Conversion from OracleDecimal to .NET)</u>	106
14-50	<u>OracleDecimal Properties</u>	107
14-51	<u>OracleDecimal Instance Methods</u>	107
14-52	<u>OracleDecimal Static Fields</u>	114
14-53	<u>OracleDecimal Static (Comparison) Methods</u>	118
14-54	<u>OracleDecimal Static (Manipulation) Methods</u>	123
14-55	<u>OracleDecimal Static (Logarithmic) Methods</u>	138
14-56	<u>OracleDecimal Static (Trigonometric) Methods</u>	144
14-57	<u>OracleDecimal Static (Comparison) Operators</u>	150
14-58	<u>OracleDecimal Static Operators (Conversion from .NET Type to OracleDecimal)</u>	159
14-59	<u>OracleDecimal Static Operators (Conversion from OracleDecimal to .NET)</u>	164
14-60	<u>OracleDecimal Properties</u>	169
14-61	<u>OracleDecimal Instance Methods</u>	173
14-62	<u>OracleIntervalDS Constructors</u>	181

14-63	<u>OracleIntervalDS Static Fields</u>	<u>181</u>
14-64	<u>OracleIntervalDS Static Methods</u>	<u>182</u>
14-65	<u>OracleIntervalDS Static Operators</u>	<u>182</u>
14-66	<u>OracleIntervalDS Type Conversions</u>	<u>183</u>
14-67	<u>OracleIntervalDS Properties</u>	<u>183</u>
14-68	<u>OracleIntervalDS Methods</u>	<u>184</u>
14-69	<u>OracleIntervalDS Static Fields</u>	<u>189</u>
14-70	<u>OracleIntervalDS Static Methods</u>	<u>192</u>
14-71	<u>OracleIntervalDS Static Operators</u>	<u>199</u>
14-72	<u>OracleIntervalDS Type Conversions</u>	<u>207</u>
14-73	<u>OracleIntervalDS Properties</u>	<u>210</u>
14-74	<u>OracleIntervalDS Methods</u>	<u>215</u>
14-75	<u>OracleIntervalYM Constructors</u>	<u>220</u>
14-76	<u>OracleIntervalYM Static Fields</u>	<u>220</u>
14-77	<u>OracleIntervalYM Static Methods</u>	<u>221</u>
14-78	<u>OracleIntervalYM Static Operators</u>	<u>221</u>
14-79	<u>OracleIntervalYM Type Conversions</u>	<u>222</u>
14-80	<u>OracleIntervalYM Properties</u>	<u>222</u>
14-81	<u>OracleIntervalYM Methods</u>	<u>222</u>
14-82	<u>OracleIntervalYM Static Fields</u>	<u>227</u>
14-83	<u>OracleIntervalYM Static Methods</u>	<u>229</u>
14-84	<u>OracleIntervalYM Static Operators</u>	<u>236</u>
14-85	<u>OracleIntervalYM Type Conversions</u>	<u>244</u>
14-86	<u>OracleIntervalYM Properties</u>	<u>246</u>
14-87	<u>OracleIntervalYM Methods</u>	<u>250</u>
14-88	<u>OracleString Constructors</u>	<u>254</u>
14-89	<u>OracleString Static Fields</u>	<u>255</u>
14-90	<u>OracleString Static Methods</u>	<u>255</u>
14-91	<u>OracleString Static Operators</u>	<u>255</u>
14-92	<u>OracleString Type Conversions</u>	<u>256</u>
14-93	<u>OracleString Properties</u>	<u>256</u>
14-94	<u>OracleString Methods</u>	<u>256</u>
14-95	<u>OracleString Static Fields</u>	<u>262</u>
14-96	<u>OracleString Static Methods</u>	<u>263</u>
14-97	<u>OracleString Static Operators</u>	<u>268</u>
14-98	<u>OracleString Type Conversions</u>	<u>274</u>
14-99	<u>OracleString Properties</u>	<u>275</u>

14-100	<u>OracleString Methods</u>	279
14-101	<u>OracleTimeStamp Constructors</u>	285
14-102	<u>OracleTimeStamp Static Fields</u>	285
14-103	<u>OracleTimeStamp Static Methods</u>	286
14-104	<u>OracleTimeStamp Static Operators</u>	286
14-105	<u>OracleTimeStamp Static Type Conversions</u>	287
14-106	<u>OracleTimeStamp Properties</u>	287
14-107	<u>OracleTimeStamp Methods</u>	288
14-108	<u>OracleTimeStamp Static Fields</u>	296
14-109	<u>OracleTimeStamp Static Methods</u>	297
14-110	<u>OracleTimeStamp Static Operators</u>	305
14-111	<u>OracleTimeStamp Static Type Conversions</u>	315
14-112	<u>OracleTimeStamp Properties</u>	321
14-113	<u>OracleTimeStamp Methods</u>	327
14-114	<u>OracleTimeStampLTZConstructors</u>	341
14-115	<u>OracleTimeStampLTZ Static Fields</u>	341
14-116	<u>OracleTimeStampLTZ Static Methods</u>	342
14-117	<u>OracleTimeStampLTZ Static Operators</u>	342
14-118	<u>OracleTimeStampLTZ Static Type Conversions</u>	343
14-119	<u>OracleTimeStampLTZ Properties</u>	343
14-120	<u>OracleTimeStampLTZ Methods</u>	344
14-121	<u>OracleTimeStampLTZ Static Fields</u>	352
14-122	<u>OracleTimeStampLTZ Static Methods</u>	354
14-123	<u>OracleTimeStampLTZ Static Operators</u>	363
14-124	<u>OracleTimeStampLTZ Static Type Conversions</u>	373
14-125	<u>OracleTimeStampLTZ Properties</u>	379
14-126	<u>OracleTimeStampLTZ Methods</u>	385
14-127	<u>OracleTimeStampTZ Constructors</u>	399
14-128	<u>OracleTimeStampTZ Static Fields</u>	399
14-129	<u>OracleTimeStampTZ Static Methods</u>	400
14-130	<u>OracleTimeStampTZ Static Operators</u>	400
14-131	<u>OracleTimeStampTZ Static Type Conversions</u>	401
14-132	<u>OracleTimeStampTZ Properties</u>	401
14-133	<u>OracleTimeStampTZ Methods</u>	402
14-134	<u>OracleTimeStampTZ Static Fields</u>	416
14-135	<u>OracleTimeStampTZ Static Methods</u>	418
14-136	<u>OracleTimeStampTZ Static Operators</u>	426

14-137	<u>OracleTimeStampTZ Static Type Conversions</u>	436
14-138	<u>OracleTimeStampTZ Properties</u>	443
14-139	<u>OracleTimeStampTZ Methods</u>	449
14-140	<u>INullable Interface Properties</u>	464
14-141	<u>INullable Interface Properties</u>	465
15-1	<u>OracleTypeException Constructor</u>	2
15-2	<u>OracleTypeException Static Methods</u>	2
15-3	<u>OracleTypeException Properties</u>	2
15-4	<u>OracleTypeException Methods</u>	3
15-5	<u>OracleTypeException Static Methods</u>	5
15-6	<u>OracleTypeException Properties</u>	5
15-7	<u>OracleTypeException Methods</u>	7
15-8	<u>OracleNullValueException Constructors</u>	9
15-9	<u>OracleNullValueException Static Methods</u>	9
15-10	<u>OracleNullValueException Properties</u>	9
15-11	<u>OracleNullValueException Methods</u>	10
15-12	<u>OracleNullValueException Static Methods</u>	12
15-13	<u>OracleNullValueException Properties</u>	12
15-14	<u>OracleNullValueException Methods</u>	13
15-15	<u>OracleTruncateException Constructors</u>	14
15-16	<u>OracleTruncateException Static Methods</u>	14
15-17	<u>OracleTruncateException Properties</u>	14
15-18	<u>OracleTruncateException Methods</u>	15
15-19	<u>OracleTruncateException Static Methods</u>	16
15-20	<u>OracleTruncateException Properties</u>	17
15-21	<u>OracleTruncateException Methods</u>	17
16-1	<u>OracleCustomTypeMappingAttribute Constructors</u>	3
16-2	<u>OracleCustomTypeMappingAttribute Static Methods</u>	3
16-3	<u>OracleCustomTypeMappingAttribute Properties</u>	3
16-4	<u>OracleCustomTypeMappingAttribute Methods</u>	3
16-5	<u>OracleCustomTypeMappingAttribute Static Methods</u>	5
16-6	<u>OracleCustomTypeMappingAttribute Properties</u>	5
16-7	<u>OracleCustomTypeMappingAttribute Methods</u>	6
16-8	<u>OracleObjectMappingAttribute Constructors</u>	8
16-9	<u>OracleObjectMappingAttribute Static Methods</u>	8
16-10	<u>OracleObjectMappingAttribute Properties</u>	8
16-11	<u>OracleObjectMappingAttribute Methods</u>	9

16-12	<u>OracleObjectMappingAttribute Static Method</u>	<u>11</u>
16-13	<u>OracleObjectMappingAttribute Properties</u>	<u>11</u>
16-14	<u>OracleObjectMappingAttribute Methods</u>	<u>13</u>
16-15	<u>OracleArrayMappingAttribute Constructors</u>	<u>14</u>
16-16	<u>OracleArrayMappingAttribute Static Methods</u>	<u>15</u>
16-17	<u>OracleArrayMappingAttribute Properties</u>	<u>15</u>
16-18	<u>OracleArrayMappingAttribute Methods</u>	<u>15</u>
16-19	<u>OracleArrayMappingAttribute Static Methods</u>	<u>16</u>
16-20	<u>OracleArrayMappingAttribute Properties</u>	<u>17</u>
16-21	<u>OracleArrayMappingAttribute Methods</u>	<u>17</u>
16-22	<u>IOracleCustomType Interface Methods</u>	<u>19</u>
16-23	<u>IOracleCustomType Interface Methods: Overload list</u>	<u>19</u>
16-24	<u>IOracleCustomTypeFactory Interface Methods</u>	<u>24</u>
16-25	<u>IOracleCustomTypeFactory Interface Methods</u>	<u>24</u>
16-26	<u>IOracleArrayTypeFactory Interface Methods</u>	<u>26</u>
16-27	<u>IOracleArrayTypeFactory Interface Methods</u>	<u>26</u>
16-28	<u>OracleUdt Static Methods</u>	<u>29</u>
16-29	<u>OracleUdt Static Methods</u>	<u>29</u>
16-30	<u>OracleRef Constructors</u>	<u>55</u>
16-31	<u>OracleRef Static Fields</u>	<u>56</u>
16-32	<u>OracleRef Static Methods</u>	<u>56</u>
16-33	<u>OracleRef Instance Properties</u>	<u>56</u>
16-34	<u>OracleRef Instance Methods</u>	<u>56</u>
16-35	<u>OracleRef Static Fields</u>	<u>60</u>
16-36	<u>OracleRef Static Methods</u>	<u>60</u>
16-37	<u>OracleRef Instance Properties</u>	<u>61</u>
16-38	<u>OracleRef Instance Methods</u>	<u>65</u>
16-39	<u>OracleUdtFetchOption Enumeration Values</u>	<u>77</u>
16-40	<u>OracleUdtStatus Enumeration Values</u>	<u>77</u>
17-1	<u>OracleBulkCopy Constructors</u>	<u>2</u>
17-2	<u>OracleBulkCopy Properties</u>	<u>3</u>
17-3	<u>OracleBulkCopy Public Methods</u>	<u>3</u>
17-4	<u>OracleBulkCopy Events</u>	<u>3</u>
17-5	<u>OracleBulkCopy Properties</u>	<u>8</u>
17-6	<u>OracleBulkCopy Public Methods</u>	<u>14</u>
17-7	<u>OracleBulkCopy Events</u>	<u>20</u>
17-8	<u>OracleBulkCopyColumnMapping Constructors</u>	<u>22</u>

17-9	<u>OracleBulkCopyColumnMapping Method</u>	<u>22</u>
17-10	<u>OracleBulkCopyColumnMapping Properties</u>	<u>23</u>
17-11	<u>OracleBulkCopyColumnMapping Method</u>	<u>26</u>
17-12	<u>OracleBulkCopyColumnMapping Properties</u>	<u>27</u>
17-13	<u>OracleBulkCopyColumnMappingCollection Properties</u>	<u>32</u>
17-14	<u>OracleBulkCopyColumnMappingCollection Public Methods</u>	<u>32</u>
17-15	<u>OracleBulkCopyColumnMappingCollection Properties</u>	<u>32</u>
17-16	<u>OracleBulkCopyColumnMappingCollection Public Methods</u>	<u>33</u>
17-17	<u>OracleBulkCopyOptions Enumeration Members</u>	<u>43</u>
17-18	<u>OracleRowsCopiedEventArgs Constructors</u>	<u>48</u>
17-19	<u>OracleRowsCopiedEventArgs Properties</u>	<u>48</u>
17-20	<u>OracleRowsCopiedEventArgs Properties</u>	<u>50</u>
18-1	<u>OracleDataProviderInstrumentationOptions Properties</u>	<u>2</u>
18-2	<u>OraclePoolNameCollection Methods</u>	<u>10</u>
19-1	<u>OciTokenAuthentication Constructors</u>	<u>2</u>
19-2	<u>OciTokenAuthentication Properties</u>	<u>2</u>
19-3	<u>OciTokenAuthentication Properties</u>	<u>3</u>
20-1	<u>AzureTokenAuthentication Constructors</u>	<u>2</u>
20-2	<u>AzureTokenAuthentication Properties</u>	<u>2</u>
20-3	<u>AzureTokenAuthentication Properties</u>	<u>4</u>
21-1	<u>OracleCollection Properties</u>	<u>4</u>
21-2	<u>OracleCollection Public Methods</u>	<u>5</u>
21-3	<u>OracleCollectionOptions Properties</u>	<u>17</u>
21-4	<u>OracleDynamicCollection Properties</u>	<u>22</u>
21-5	<u>OracleDynamicCollection Public Methods</u>	<u>22</u>
21-6	<u>OracleServiceCollectionExtensions Static Methods</u>	<u>34</u>
21-7	<u>OracleVectorStore Public Methods</u>	<u>46</u>
21-8	<u>OracleVectorStoreOptions Properties</u>	<u>52</u>
A-1	<u>MetaDataCollections</u>	<u>A-1</u>
A-2	<u>DataSource Information</u>	<u>A-2</u>
A-3	<u>Data Types</u>	<u>A-3</u>
A-4	<u>Restrictions</u>	<u>A-5</u>
A-5	<u>ReservedWords</u>	<u>A-5</u>
A-6	<u>Tables</u>	<u>A-6</u>
A-7	<u>Columns</u>	<u>A-7</u>
A-8	<u>Views</u>	<u>A-7</u>
A-9	<u>XMLSchema</u>	<u>A-8</u>

A-10	<u>Users</u>	<u>A-8</u>
A-11	<u>Synonyms</u>	<u>A-8</u>
A-12	<u>Sequences</u>	<u>A-9</u>
A-13	<u>Functions</u>	<u>A-9</u>
A-14	<u>Procedures</u>	<u>A-10</u>
A-15	<u>ProcedureParameters</u>	<u>A-11</u>
A-16	<u>Arguments</u>	<u>A-12</u>
A-17	<u>Packages</u>	<u>A-13</u>
A-18	<u>PackageBodies</u>	<u>A-13</u>
A-19	<u>JavaClasses</u>	<u>A-14</u>
A-20	<u>Indexes</u>	<u>A-15</u>
A-21	<u>IndexColumns</u>	<u>A-17</u>
A-22	<u>PrimaryKeys</u>	<u>A-18</u>
A-23	<u>ForeignKeys</u>	<u>A-18</u>
A-24	<u>ForeignKeyColumns</u>	<u>A-19</u>
A-25	<u>UniqueKeys</u>	<u>A-20</u>
B-1	<u>Mapping of Aggregate Canonical Functions and Oracle Functions</u>	<u>B-1</u>
B-2	<u>Mapping of Math Canonical Functions and Oracle Functions</u>	<u>B-1</u>
B-3	<u>Mapping of String Canonical Functions and Oracle Functions</u>	<u>B-2</u>
B-4	<u>Mapping of Date And Time Canonical Functions and Oracle Functions</u>	<u>B-2</u>
B-5	<u>Mapping of Bitwise Canonical Functions and Oracle Functions</u>	<u>B-4</u>
B-6	<u>Mapping of Other Canonical Functions and Oracle Functions</u>	<u>B-4</u>

Preface

This document is your primary source of introductory, installation, postinstallation configuration, and usage information for Oracle Data Provider for .NET.

Oracle Data Provider for .NET is an implementation of the Microsoft ADO.NET interface.

This Preface contains these topics:

- [Audience](#)
- [Documentation Accessibility](#)
- [Related Documents](#)
- [Passwords in Code Examples](#)
- [Conventions](#)

Audience

Oracle Data Provider for .NET Developer's Guide is intended for programmers who are developing applications to access an Oracle database using Oracle Data Provider for .NET. This documentation is also valuable to systems analysts, project managers, and others interested in the development of database applications.

To use this document, you must be familiar with Microsoft .NET Framework classes and ADO.NET and have a working knowledge of application programming using Microsoft C#, Visual Basic .NET, or another .NET language.

Although the examples in the documentation and the samples in the sample directory are written in C#, developers can use these examples as models for writing code in other .NET languages.

Users should also be familiar with the use of Structured Query Language (SQL) to access information in relational database systems.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Related Documents

For more information, see these Oracle resources:

- *Oracle Database Installation Guide for Microsoft Windows*
- *Oracle Database Release Notes*
- *Oracle Database Administrator's Guide*
- *Oracle Database Development Guide*
- *Oracle Database SecureFiles and Large Objects Developer's Guide*
- *Oracle Real Application Clusters Administration and Deployment Guide*
- *Oracle Database New Features Guide*
- *Oracle Database Concepts*
- *Oracle Database Reference*
- *Oracle Database Object-Relational Developer's Guide*
- *Oracle Database SQL Language Reference*
- *Oracle Database Net Services Administrator's Guide*
- *Oracle Database Net Services Reference*
- *Oracle Call Interface Programmer's Guide*
- *Oracle Services for Microsoft Transaction Server Developer's Guide for Microsoft Windows*
- *Oracle Database Globalization Support Guide*
- *Oracle XML DB Developer's Guide*
- *Oracle XML Developer's Kit Programmer's Guide*
- *Oracle Database Security Guide*
- *Oracle Spatial Developer's Guide*
- *Oracle Data Guard Concepts and Administration*

To download free release notes, installation documentation, white papers, or other collateral, please visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

<http://www.oracle.com/technetwork/index.html>

For additional information, see:

<https://learn.microsoft.com/en-us/docs/>

Passwords in Code Examples

For simplicity in demonstrating this product, code examples do not perform the password management techniques that a deployed system normally uses. In a production environment, follow the Oracle Database password management guidelines, and disable any sample accounts. See *Oracle Database Security Guide* for password management guidelines and other security recommendations.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Changes in This Release for Oracle Data Provider for .NET

This section describes new features added for each ODP.NET release. A summary of these new features by release are also available on [Oracle's Web site](#).

- [Changes in Oracle Data Provider for .NET Release 23.26.0](#)
- [Changes in Oracle Data Provider for .NET Release 23.9](#)
- [Changes in Oracle Data Provider for .NET Release 23.8](#)
- [Changes in Oracle Data Provider for .NET Release 23.7](#)
- [Changes in Oracle Data Provider for .NET Release 23.6.1](#)
- [Changes in Oracle Data Provider for .NET Release 23.6](#)
- [Changes in Oracle Data Provider for .NET Release 23.5.1](#)
- [Changes in Oracle Data Provider for .NET Release 23.5](#)
- [Changes in Oracle Data Provider for .NET Release 23.3](#)
- [Changes in Oracle Data Provider for .NET Release 21c \(21.12\)](#)
- [Changes in Oracle Data Provider for .NET Release 21c \(21.8\)](#)
- [Changes in Oracle Data Provider for .NET Release 21c \(21.7\)](#)
- [Changes in Oracle Data Provider for .NET Release 21c \(21.6.1\)](#)
- [Changes in Oracle Data Provider for .NET Release 21c \(21.5\)](#)
- [Changes in Oracle Data Provider for .NET Release 21c \(21.4.1\)](#)
- [Changes in Oracle Data Provider for .NET Release 21c \(21.4\)](#)
- [Changes in Oracle Data Provider for .NET Release 21c \(21.3\)](#)
- [Changes in Oracle Data Provider for .NET \(21.1\)](#)
- [Changes in Oracle Data Provider for .NET \(19.15.1\)](#)
- [Changes in Oracle Data Provider for .NET \(19.10\)](#)
- [Changes in Oracle Data Provider for .NET \(19.9\)](#)
- [Changes in Oracle Data Provider for .NET ODAC Release 19c \(19.3.2\)](#)
- [Changes in Oracle Data Provider for .NET \(19.7\)](#)
- [Changes in Oracle Data Provider for .NET \(19.6\)](#)
- [Changes in Oracle Data Provider for .NET \(19.5\)](#)
- [Changes in Oracle Data Provider for .NET in ODAC Release 19c \(19.3\)](#)
- [Changes in Oracle Data Provider for .NET Release 19c \(19.3\)](#)

Changes in Oracle Data Provider for .NET Release 23.26.0

The following are the changes in Oracle Data Provider for .NET for Release 23.26.0.

New Features

The following features are new in this release:

- .NET 10 Runtime

ODP.NET Core is certified for .NET 10.

See Also

[System Requirements](#)

- Entity Framework Core 10

ODP.NET Entity Framework Core now supports Entity Framework Core 10 with Oracle databases. ODP.NET EF Core NuGet package 10.23.2600 or higher is required.

See Also

[Oracle Entity Framework Core 10 Features](#)

- Managed ODP.NET and ODP.NET Core – Upgrading ODP.NET Time Zone File

ODP.NET now allows an app to specify a newer time zone file if the ODP.NET default uses an older version. This feature eliminates errors from using outdated time zone files.

See Also

OracleConfiguration.[TimeZoneFileLocation](#)

OracleConnection.[DatabaseTimeZoneFileVersion](#)

OracleConnection.[ClientTimeZoneFileVersion](#)

- Managed ODP.NET and ODP.NET Core – OpenTelemetry Rows Fetched per Round Trip

ODP.NET OpenTelemetry now returns the number of rows returned per database roundtrip for ODP.NET SELECT statements in a new `db.response.returned_rows` tag.

This tag enhances visibility into ODP.NET activity and performance.

See Also

[OpenTelemetry](#)

- Managed ODP.NET and ODP.NET Core – Application Continuity Maximum Calls Per Request Boundary

In Application Continuity, ODP.NET stores all the driver calls that occur between request boundaries, generally when connections are checked out and in. Having this call record is what makes session restoration possible after a failover. For long-running connections, more calls to restore mean more memory is necessary to retain all the changes between request boundaries. ODP.NET allows you to disable session replay when this memory usage becomes too high by setting a maximum number of calls to retain between request boundaries.

If you have long-running connections using Application Continuity that consume too much memory for replay, you can free memory from the connections with the most calls by setting the `OracleConfiguration.MaxCallsPerRequestBoundary` property to a lower value.

See Also

OracleConfiguration [MaxCallsPerRequestBoundary](#) property

- Raft Replication

All ODP.NET provider types now support Oracle Globally Distributed Database Raft replication for fault-tolerant replication with transactional support.

ODP.NET apps will then have automatic failover with no data loss while using distributed databases and Oracle Data Guard. No ODP.NET code changes are necessary to use Raft replication.

See Also

ODP.NET [Oracle Globally Distributed Database](#)

Using Raft Replication in Oracle Globally Distributed Database

Changes in Oracle Data Provider for .NET Release 23.9

The following are the changes in Oracle Data Provider for .NET for Release 23.9.

New Features

The following features are new in this release:

- ODP.NET Core – Distributed Transactions on Windows

With a fully managed System.Transactions library now available, ODP.NET Core now supports distributed transactions on Windows. In-doubt transaction recovery occurs via the Oracle Services for MTS recovery service.

Distributed transactions apps running in .NET Framework can be ported to .NET runtime. New .NET distributed transaction apps can be developed with .NET 8 and higher.

See Also

[Distributed Transactions](#)

- Managed ODP.NET and ODP.NET Core – Binary Sparse Vector Data Types for AI Apps
ODP.NET enhances its sparse vector data type support by adding binary numeric format. ODP.NET sparse vectors can now be represented in all four numeric formats: BINARY, FLOAT32, FLOAT64, and INT8.

The BINARY format advantages are:

- Its vector storage footprint can be reduced by a factor of 32 compared to FLOAT32.
- Distance computations between two vectors are up to 40 times faster.

Its disadvantage is possible accuracy loss, though that loss is often not substantial.

See Also

[Artificial Intelligence Vectors and Semantic Search](#)

- Managed ODP.NET and ODP.NET Core – Bulk Copy Vector Data Types
ODP.NET now supports bulk copying vector data types into the database.
Apps can now efficiently load AI and vector data sets for use in Oracle database.

See Also

[Bulk Copy](#)

- Managed ODP.NET and ODP.NET Core – ODP.NET OpenTelemetry Round Trip Count and Duration Activity Tags
ODP.NET OpenTelemetry now returns the total roundtrips and the duration of individual ODP.NET method calls.

These tags enhance visibility into ODP.NET activity and performance.

See Also

[OpenTelemetry](#)

- ODP.NET Core – PEM Files for Securing ODP.NET Apps
ODP.NET now supports PEM files for storing client security certificates.
This feature gives apps another option to store credentials and certificates.

See Also

[Mutual TLS/SSL Certificate Selection](#)

- Managed ODP.NET – Kerberos.NET
Managed ODP.NET has the option to utilize the Kerberos.NET library for Kerberos authentication. ODP.NET Core already uses Kerberos.NET.
Developers can now standardize on Kerberos.NET for both their ODP.NET managed and core apps. Managed ODP.NET programmers can choose between MIT Kerberos or

Kerberos.NET. Oracle recommends customers select Kerberos.NET as it is a fully managed solution built specifically for .NET.

① See Also

[Using Kerberos](#)

- SHA2-based Thumbprint for Certificate Selection

ODP.NET can now select certificates via thumbprint using SHA2 algorithms in addition to existing SHA1 support.

SHA2 is generally considered more secure than SHA1 and provides another security option for ODP.NET apps.

① See Also

OracleConfiguration.[SSLCertificateThumbprint](#)

OracleConnection.[SSLCertificateThumbprint](#)

- Easier TLS Version Configuration

ODP.NET can specify a minimum TLS version by appending the plus (+) sign to the property version value, such as 1.2+ or TLSv1.2+. ODP.NET will automatically use the highest TLS version the Oracle database server supports. This property can be set in the TLSVersion or SSLVersion properties.

To exclude specific higher TLS versions from use, developers set the TLSExcludeVersion property to identify the excluded versions.

These new capabilities make it easier to upgrade and specify newer TLS versions when they become available on the database server.

① See Also

[OracleConfiguration Secure Connection Properties](#)

[OracleConnection Properties](#)

- ODP.NET Core – More OracleDataSource and OracleDataSourceBuilder Properties

OracleDataSource and OracleDataSourceBuilder classes support the new properties, RetryCount and RetryDelay.

In addition, OracleDataSource supports the DatabaseCharset, DatabaseNCharset, and MaxIdentifierLength properties.

① See Also

[OracleDataSource Properties](#)

OracleDataSourceBuilder [Methods, Properties, and Events in Common with OracleConnection or OracleDataSource](#)

- Managed ODP.NET and ODP.NET Core – Configurable Connection String URL Expiration Grace Period

Applications set the time to keep connection string URLs cached. Once they expire, ODP.NET attempts to refresh the connection string values. If it can not, then it continues to use the old connection string URL to open new connections. The length of this time can now be set using the `config_time_to_live_grace_period` property.

Apps can determine the grace period expired connection string URLs can continue to be used.

 See Also

[Connection URL Caching](#)

Changes in Oracle Data Provider for .NET Release 23.8

The following are the changes in Oracle Data Provider for .NET for Release 23.8.

New Features

The following features are new in this release:

- ODP.NET Core – `OracleDataSource` and `OracleDataSourceBuilder`

With the new `OracleDataSource` and `OracleDataSourceBuilder` classes, ODP.NET Core supports ADO.NET `DbDataSource` from which apps can retrieve connections and execute commands directly. The class encapsulates all Oracle database connection configuration.

`OracleDataSource` is portable and can be registered in dependency injection as a connection factory. It can be used in provider agnostic code to connect to Oracle databases.

`OracleDataSourceBuilder` instantiates an immutable `OracleDataSource` class.

 See Also

[DbDataSource](#)

- Managed ODP.NET and ODP.NET Core – Sparse and Dense Vector Data Types for AI Apps

ODP.NET now supports new vector data types for storing sparse and dense data sets. Sparse vectors typically have a large number of dimensions and few non-zero dimension values. Dense vectors are better suited when nearly every dimension stores a non-zero value.

Because sparse vectors only store non-zero values, they are more storage and memory efficient and ODP.NET can process them faster than dense vectors when there are few non-zero dimensions. That makes them better for some AI applications, such as text retrieval systems in which an inputted phrase identifies relevant matching documents.

Dense vectors are better for semantic search when the word meaning nuances are important to identify matches.

ODP.NET EF Core 9 and higher now support sparse vectors in addition to its existing support of dense vectors.

See Also

[Artificial Intelligence Vectors and Semantic Search](#)

- EF Core – Scaffolding Data Type Mapping Customization Templates

Oracle EF Core scaffolding of entity properties from database columns can now be customized using custom reverse engineering templates when developers prefer an alternative set of .NET data type mappings than the default.

Developers can flexibly choose a more complementary set of .NET data type mappings for their database columns, including CLR types that store all possible values of the mapped Oracle Database data types.

See Also

[Scaffolding Or Reverse Engineering](#)

- Managed ODP.NET and ODP.NET Core – PKCS #12 Wallet Files and Secure External Password Store

ODP.NET now supports using PKCS #12 wallet files and Secure External Password Store. PKCS #12 is a popular file format that stores security credentials and certificates. These files typically use the .p12 file extension.

This feature gives ODP.NET apps another option for storing credential and certificate file storage.

- Managed ODP.NET and ODP.NET Core – Protect ODP.NET Code from SQL Injection Attacks

ODP.NET developers write secure code using bind variables to limit SQL injection attacks. However, it's not always possible to use bind variables. ODP.NET adds new methods to sanitize the input literal and database identifier values to protect against SQL injection attacks.

Through some simple code modifications, developers can secure their SQL literal and identifier usage easily and automatically.

See Also

[Securing Identifiers and Literals from SQL Injection Attacks](#)

[OracleDBMSAssert Class](#)

- Automatic Connection Recovery

ODP.NET can now automatically detect invalid connections resulting from planned or unplanned outages. After detecting an invalid session, ODP.NET fails over by creating a new session to an available database instance. The end user can conduct their work without encountering an error.

This feature makes apps more resilient and highly available without developers have to add or maintain code to do so.

① See Also[Automatic Connection Recovery](#)

- Prevent Application State Leaks Between Database Sessions

When an ODP.NET session returns to the connection pool, the database session state is partially reset preventing much of the state from leaking to the next connection's state. This feature is enabled by setting the RESET_STATE database service attribute to LEVEL1.

This enhancement ensures all newly dispensed connections have a more consistent session state with fewer artifacts which can occur after a failover event.

① See Also[Reset Session State to Limit Application State Leaks](#)

- Managed ODP.NET and ODP.NET Core – Bulk Copy Boolean Data Types

Managed ODP.NET and ODP.NET Core provider types now support Oracle database Boolean data types with bulk copy.

Apps can now efficiently load data sets that contain Boolean data.

① See Also[Bulk Copy](#)

- Managed ODP.NET and ODP.NET Core – Bulk Copy Index Check Enforcement

ODP.NET bulk copy can enforce index checks when this option is enabled. When an index-related error occurs, no rows are loaded and indexes remain unchanged.

Apps can now abort data loads unless constraints are kept in an usable state.

① See Also[OracleBulkCopyOptions Enumeration](#)

Changes in Oracle Data Provider for .NET Release 23.7

The following are the changes in Oracle Data Provider for .NET for Release 23.7.

New Features

The following features are new in this release:

- ODP.NET Core -- Alpine Linux and macOS

ODP.NET Core is certified with Alpine Linux and macOS (ARM64).

① See Also

[System Requirements](#)

- Managed ODP.NET and ODP.NET Core – Sessionless Transaction

ODP.NET sessionless transactions enable apps to start a transaction, suspend the transaction, and then continue the work in another database session before finally committing.

With sessionless transactions, transactions can span multiple sessions for more app development flexibility. Programmers have more options available, allowing them to select more efficient, easier to manage, and faster running app implementations.

① See Also

[Sessionless Transactions](#)

- Managed ODP.NET and ODP.NET Core – Set Rows to Fetch Per Round Trip

ODP.NET apps can now set the number of rows to fetch per database round trip using the `RowsToFetchPerRoundTrip` property available in the .NET configuration file, `OracleConfiguration`, `OracleConnection`, `OracleCommand`, `OracleDataReader`, and `OracleRefCursor` classes.

This setting is easier for developers to use when they know exactly the number of rows they wish to fetch for each round trip.

① See Also

[Controlling the Number of Rows Fetched in One Database Round-Trip](#)

- Managed ODP.NET and ODP.NET Core – Set Database Role on Connections

Database roles are created by users, usually administrators, to group multiple privileges or other roles together. ODP.NET can set database roles programmatically on connections it creates and those retrieved from the connection pool. The connection can then use the role's privileges.

This feature makes assigning database roles to ODP.NET database connections straightforward and simple.

① See Also

[OracleRoleCollection Class](#)

`OracleConnection` [Roles](#)

- Managed ODP.NET and ODP.NET Core – Customize Each Connection's Encryption and Data Integrity Settings

ODP.NET can now set encryption and data integrity properties for each connection at the `OracleConnection` level instead of at only the app level with `OracleConfiguration`. App

connections may not all use the same security settings. Each connection can have different security settings if needed.

This feature allows apps to customize and individualize their connection security settings when business and technical requirements dictate.

See Also

[OracleConnection Properties](#)

- Managed ODP.NET and ODP.NET Core – Server Name Indication

Server Name Indication (SNI) is a TLS extension normally used to send the hostname in the client hello message. ODP.NET now supports SNI.

SNI is used by Oracle to pass the relevant connect_data information, such as the service name, clients send when connecting to the database server. On the listener side, the requested connect data is obtained without performing TLS negotiation and consuming the client hello packet. With this information, an appropriate handler is selected. The listener can proceed with spawning dedicated server process.

SNI improves resource usage, scalability, security, and ease of management.

See Also

OracleConnection [UseSNI](#)

OracleConfiguration [UseSNI](#)

- Managed ODP.NET and ODP.NET Core – Enhanced Password Security with OracleOpaqueString

.NET SecureString class is more secure than the String class, but still has some security weaknesses. OracleOpaqueString class securely obfuscates string data, such as passwords, without SecureString's limitations. ODP.NET APIs that use SecureString now have the same APIs that use OracleOpaqueString.

Oracle recommends using OracleOpaqueString for more robust security instead of SecureString.

See Also

[Securing Passwords with OracleOpaqueString](#)

- Managed ODP.NET and ODP.NET Core - Sharding Split Partitionset

ODP.NET supports sharding split partitionset events in core and managed drivers. A split partitionset is an operation on the sharding database performed when moving data within a specified super sharding key to a different shardspace.

ODP.NET connection pools receive database events about data in a chunk being split and moved across these partition sets. The provider updates its sharding topology appropriately and transparently so that end users are dispensed the correct connection to use.

[① See Also](#)[Sharding Split Partitionset](#)

Changes in Oracle Data Provider for .NET Release 23.6.1

The following are the changes in Oracle Data Provider for .NET for Release 23.6.1.

New Features

The following feature is new in this release:

- .NET 9 Runtime

ODP.NET Core is certified for .NET 9.

[① See Also](#)[System Requirements](#)

- Managed ODP.NET and ODP.NET Core – Connection Configuration Restriction

ODP.NET can now be configured to restrict connection string Data Source attribute settings at a global level when using full connect descriptors, Easy Connect (Plus), or TNS net service names.

Independent software vendors and application administrators have an easy way to control which connection settings and options are allowed. This capability makes securing application connectivity simpler.

[① See Also](#)[Connection Configuration Restriction](#)

Changes in Oracle Data Provider for .NET Release 23.6

The following are the changes in Oracle Data Provider for .NET for Release 23.6.

New Features

The following features are new in this release:

- Entity Framework Core 9

ODP.NET Entity Framework (EF) Core now supports Entity Framework Core 9 with Oracle databases. ODP.NET EF Core NuGet package 9.23.60 or higher is required.

① See Also

[Oracle Entity Framework Core 9 Features](#)

- ODP.NET Core -- Kerberos

ODP.NET Core now supports Kerberos, a network authentication service for security in distributed environments. ODP.NET applications can use Kerberos for single sign-on and centralized user authentication.

① See Also

[Using Kerberos](#)

- Managed ODP.NET and ODP.NET Core – OpenTelemetry enhancements

OpenTelemetry has been enhanced to publish the ODP.NET NuGet version as the activity source version, ODP.NET executed statement texts, and to redact traced command text to hide literal values for security reasons.

These new features provide more trace options and capabilities for ODP.NET OpenTelemetry.

① See Also

[OpenTelemetry](#)

- Managed ODP.NET and ODP.NET Core – Vector BINARY Numeric Format

ODP.NET developers can use the BINARY dimension format for vectors.

The BINARY numeric format is 32 times smaller than the FLOAT32 format and distance computations between two vectors are up to 40 times faster.

There is a potential loss of accuracy with format; however, the loss is often not substantial.

① See Also

[Artificial Intelligence Vectors and Semantic Search](#)

- Managed ODP.NET and ODP.NET Core – Easier proxy server setup for cloud deployments

More ODP.NET apps are being deployed with cloud databases that are behind firewalls. This setup requires using web proxy servers in the Oracle database connection address. New OracleConnection properties (AutoProxy, HttpsProxy, HttpsProxyPort, SSLServerDNMatch, SSLServerCertDN, and SSLVersion) have been introduced to make it easier to use web proxy servers.

These interfaces allow ODP.NET to detect or set the proxy so that proxy setup is simpler.

See Also

[OracleConnection Class](#)

- Managed ODP.NET and ODP.NET Core – IsFloatingPointNumber Schema Table Column
This new column in the ADO.NET SchemaTable indicates whether the result set column data type is a floating-point number or not.
It allows apps to easily distinguish whether a column stores floating-point numbers or some other type, such as fixed-point numbers.

See Also

OracleDataReader [GetSchemaTable](#)

- Managed ODP.NET and ODP.NET Core – Disallow UNC Paths
Apps can disallow ODP.NET access to configuration files with Universal Naming Convention (UNC) location paths.
Server Message Block coercion attacks use UNC paths to have a target machine authenticate an attacker's computer. If successful, then the attacker copies that authentication to gain access to another system. Disabling UNC paths protects ODP.NET apps from this attack.

See Also

OracleConfiguration [AllowUncPaths](#)

- Managed ODP.NET and ODP.NET Core – User-Created Wallet Stored Client Secret for Azure Service Principal Authentication
ODP.NET now allows Azure Service Principal authentication flows using a client secret stored in a user-created wallet in addition to the existing methods of providing the client secret directly and client certificate.
This new option securely stores the client secret and provides more flexibility when using Azure Service Principal with ODP.NET.

See Also

[Azure Centralized Configuration Provider](#)

- Managed ODP.NET and ODP.NET Core – Disable Double Encryption and Double Data Integrity
When TCPS is enabled along with Oracle Native Network Encryption (NNE) and/or data integrity, ODP.NET can be configured to use TCPS only.
This feature allows apps to avoid inadvertently enabling double encryption and double checksumming, which can lower performance.

① See Also

OracleConfiguration [SqlNetIgnoreANOEncryptionForTCPS](#)

Changes in Oracle Data Provider for .NET Release 23.5.1

The following are the changes in Oracle Data Provider for .NET for Release 23.5.1.

New Features

The following feature is new in this release:

- Managed ODP.NET and ODP.NET Core - Eliminate Exception Using Transparent Application Failover SELECT Mode

Managed ODP.NET and ODP.NET Core 23.5.1 introduced TAF session mode support. TAF SELECT mode is not yet supported.

ODP.NET 23.5.1 has been enhanced so that no error occurs when TAF SELECT mode is configured. ODP.NET will use TAF session mode instead.

Changes in Oracle Data Provider for .NET Release 23.5

The following are the changes in Oracle Data Provider for .NET for Release 23.5.

New Features

The following features are new in this release:

- Managed ODP.NET and ODP.NET Core - REF CURSOR column type

ODP.NET can now return REF CURSOR type column in a select list to an `OracleDataReader`, `OracleRefCursor`, or as JSON-formatted .NET string or `OracleString` result set.

This feature improves flexibility and adds another way to consume Oracle result sets.

① See Also

`OracleDataReader`.[GetOracleDataReader](#)

`OracleDataReader`.[GetOracleRefCursor](#)

- Managed ODP.NET and ODP.NET Core – Globally Suppress GetDecimal Invalid Cast Exception

The `SuppressGetDecimalInvalidCastException` property has been added to `OracleConfiguration` and `OracleConnection` classes. Previously, they were available to only to `OracleDataReader` and `OracleDataAdapter` classes. When enabled and `GetDecimal` is called, ODP.NET suppresses the `InvalidCastException` and returns a rounded-off 28 or 29 precision .NET decimal value, that represents the Oracle NUMBER.

This feature allows the exception suppression to be set at either the application level (`OracleConfiguration`) or connection level (`OracleConnection`), which improves ease of

use. All of the settings also enable suppressing the exception when UDTs and `OracleParameters` retrieve large numbers.

See Also

`OracleConfiguration`.[SuppressGetDecimalInvalidCastException](#)
`OracleConnection`.[SuppressGetDecimalInvalidCastException](#)

- Managed ODP.NET and ODP.NET Core – ODP.NET Product Information and Application Metadata

ODP.NET adds new properties that indicate the provider type (core or managed), NuGet version, and allow applications to assign a program name to identify the client app in the database's `PROGRAM` column in `V$SESSION`.

These new properties provide a systematic way for database administrators and application support personnel to identify the ODP.NET provider type, version, and application name.

See Also

`OracleConnection`.[ProviderName](#)
`OracleConnection`.[ProviderNuGetVersion](#)
`OracleConfiguration`.[Program](#)

- Managed ODP.NET and ODP.NET Core – OCI Object Storage Change Notification for Connection Information

Since connection information retrieval takes longer from a cloud resource than local disk, ODP.NET caches connection information. If that information is updated, cache updates can occur using a pull model (polling) or automated push model (change notification). Change notification for ODP.NET connection attributes stored in Azure was previously introduced. This new release introduces the same capability for Oracle Cloud Infrastructure Object Storage.

Change notification provides an automated way to get fast updates to connection properties changes without having to sacrifice the performance of storing the information locally.

See Also

[Connection URL Caching](#)

- Managed ODP.NET and ODP.NET Core – Bulk Copy Notification After All Rows Processed

ODP.NET bulk copy can now send a notification after all rows have been processed when the bulk copy operation has completed. This feature is enabled using `OracleBulkCopyOptions.NotifyAllRowsProcessed`. The number of rows is returned via the `OracleRowsCopiedEventArgs.RowsCopied` property in the notification.

By using this property, applications can now determine the total number of rows processed when their bulk copy operations complete.

① See Also[OracleBulkCopyOptions Enumeration](#)[OracleRowsCopiedEventArgs.RowsCopied](#)

- Managed ODP.NET and ODP.NET Core – Easy Connect Plus Address Lists

ODP.NET incorporates address list support in Easy Connect Plus configurations.

With address lists, multiple lists of addresses, each with its own characteristics can be configured for ODP.NET connections.

① See Also[Easy Connect and Easy Connect Plus Naming Methods](#)

- Managed ODP.NET and ODP.NET Core – Microsoft identity platform tokens v2.0 enhancement

ODP.NET supports Microsoft identity platform tokens v1.0 for Microsoft Entra ID-only applications and v2.0 for applications that support consumer accounts. ODP.NET 23.5 and higher check the roles (roles) and user principal name (UPN) claims in v2.0 tokens to verify at least one is present. If both claims are missing, then the provider will report an error.

① See Also[Vault Wallets and Tokens](#)

- Managed ODP.NET and ODP.NET Core – OpenTelemetry Activity Tag for rows affected by SQL

ODP.NET OpenTelemetry now returns the row count affected by SQL DML (INSERT, UPDATE, and DELETE only) when using `ExecuteNonQuery*`, `ExecuteReader*`, or `ExecuteScalar*` methods.

① See Also[OpenTelemetry](#)

- Unmanaged ODP.NET – Dynamic loading of `System.Text.Json` DLL and its dependencies

Unmanaged ODP.NET adds support for dynamic loading of the `System.Text.Json.dll` and its dependencies if they are not loaded by the application or a different version is loaded. This enhancement is for unmanaged ODP.NET's token-based authentication feature used by the applications, such as Power BI.

Changes in Oracle Data Provider for .NET Release 23.3

The following are the changes in Oracle Data Provider for .NET for Release 23.3.

New Features

The following features are new in this release:

- ODP.NET: AI Vector Search

You can now leverage semantic search capabilities using Artificial Intelligence (AI) Vector Search. These capabilities include a new vector data type, vector indexes, and vector search SQL operators that allow the database to store semantic document content, images, and other unstructured data as vectors and then run fast performing similarity queries. This allows the database to better understand user intent and the search context to find similar matches, rather than only find exact matches.

AI Vector Search provides your business with the capability to leverage Generative AI constructs, specifically Retrieval Augmented Generation (RAG), for responding to natural language questions. RAG effectively merges large language models (LLMs) with your private business data, preserving data privacy while enhancing response accuracy. This functionality allows you to unlock new opportunities for growth with AI.

See Also

[Artificial Intelligence Vectors and Semantic Search](#)

- Managed ODP.NET and ODP.NET Core - Asynchronous Programming

ODP.NET supports the .NET Task Asynchronous Programming (TAP) model with the core and managed drivers. With support for TAP and the `async` and `await` keywords, ODP.NET data access operations are more responsive and easier to develop for asynchronicity. This feature includes support for Entity Framework Core and Entity Framework asynchronous query and save.

See Also

[Asynchronous Programming and Pipelining](#)

- Managed ODP.NET and ODP.NET Core - Pipelining

ODP.NET core and managed drivers support pipelining for its database communication. It allows subsequent database requests to be sent and queued transparently even while ODP.NET awaits a database response. Pipelining improves overall app performance and allows database resources to be used more effectively. ODP.NET does not need to wait for the database to respond from previous requests before submitting subsequent requests.

See Also

[Asynchronous Programming and Pipelining](#)

- Managed ODP.NET and ODP.NET Core – OpenTelemetry

OpenTelemetry is a popular open-source observability framework for instrumenting, generating, collecting, and exporting telemetry data. It provides a common specification and protocol so that multiple services can furnish a unified version of traces, metrics, and logs.

Numerous managed ODP.NET and ODP.NET Core APIs have been instrumented to support OpenTelemetry observability and standards. Developers and operators can customize the ODP.NET metrics collected with OpenTelemetry. ODP.NET OpenTelemetry includes support for automatic, dynamic, and manual instrumentation.

With OpenTelemetry support, monitoring, tracking, and analyzing how ODP.NET operations interact in cloud computing, microservices and distributed systems becomes easier using this industry standard.

① See Also

[OpenTelemetry](#)

- Managed ODP.NET and ODP.NET Core - .NET Metrics

.NET Metrics are application numerical measurements collected at regular time intervals for the purposes of monitoring and alerting about application health. In an ODP.NET setting, metrics can monitor connection statistics, such as number of ODP.NET hard connections to the database, number of active connections, or number of free connections.

ODP.NET Core and managed ODP.NET support .NET Metrics. ODP.NET metrics can be published to and analyzed by the rich and expansive toolsets integrated with OpenTelemetry and .NET Metrics, such as Grafana and Prometheus.

① See Also

[.NET Metrics](#)

- Managed ODP.NET and ODP.NET Core - Advanced Queuing and Transactional Event Queues

ODP.NET Core and managed ODP.NET now support Advanced Queuing (AQ) and Transactional Event Queues (TxEventQ) application programming interfaces (APIs) that can be used in modern applications, such as microservices. TxEventQ's highly optimized and partitioned implementation leverages the functions of Oracle AI Database so that producers and consumers can exchange messages at high throughput, by storing messages persistently, and propagate messages between queues on different databases. TxEventQ are a high performance partitioned implementation with multiple event streams per queue, while AQ is a disk-based implementation for simpler workflow use cases.

ODP.NET developers can leverage the same APIs no matter if they use TxEventQ or AQ. The APIs provide access to a robust and feature-rich message queuing systems integrated with Oracle AI Database. It can be used with web, mobile, IoT, and other data-driven and event-driven applications to stream events or communicate with each other as part of a workflow.

① See Also

[Advanced Queuing and Transactional Event Queues](#)

- Managed ODP.NET and ODP.NET Core - Application Continuity and Transparent Application Continuity

ODP.NET Core and managed drivers now support Application Continuity (AC) and Transparent Application Continuity (TAC). AC and TAC mask outages from end users and

applications by recovering the in-flight database sessions following recoverable outages, including transactions. The recovery is transparent such that the end user merely experiences a slightly delayed execution, but no perceptible outage nor error.

AC and TAC improve the user experience for both unplanned outages and planned maintenance. They enhance the fault tolerance of systems and .NET applications that use an Oracle AI Database. Developers can use AC and TAC with existing .NET apps without making any code changes.

① See Also

[ODP.NET and Application Continuity](#)

- Managed ODP.NET and ODP.NET Core - Transparent Application Failover

Oracle Transparent Application Failover (TAF) is a high availability feature that enables client apps to automatically reconnect to a secondary database instance if the connected primary instance fails or shuts down. ODP.NET Core and managed ODP.NET now support connection and basic session state TAF.

ODP.NET TAF enables apps to recover and continue operating when database downtime occurs. It requires no changes to .NET application code to use.

① See Also

[Transparent Application Failover](#)

- JSON Relational Duality

JSON Relational Duality manifests as fully updatable JSON views over relational data. Data remains stored in relational tables in a highly efficient normalized format but can be accessed by .NET applications in the form of JSON documents. All ODP.NET provider types, core, managed, and unmanaged, support using JSON Relational Duality.

Duality views provide game-changing flexibility and simplicity by overcoming the historical challenges .NET developers have faced when building applications using relational or document models.

① See Also

[Native JSON Support](#)

- Oracle Client Driver Support for SQL BOOLEAN Data Type

Oracle client drivers support fetching and binding the new B00LEAN database column. Applications can use the native database B00LEAN column data type with a native driver B00LEAN data type. This enhancement makes working with B00LEAN data types easier for developers.

- True Cache

Oracle True Cache is a read-only, in-memory, high performance SQL and key-value cache that is automatically managed and consistent. It improves application response time while reducing the load on the database server. Automatic management and consistency simplify application development; reducing developer effort and cost.

All ODP.NET provider types can transparently use Oracle True Cache without needing any code changes.

- **Faster Connection Establishment**

ODP.NET connection establishment has been optimized to reduce the number of database round trips required when connecting with Oracle AI Database 26ai and higher. Applications that upgrade experience this performance improvement transparently.

- **Managed ODP.NET and ODP.NET Core - Centralized Configuration Providers**

Managed ODP.NET and ODP.NET Core can store and retrieve application configuration data from a centralized on-premises, cloud infrastructure, Oracle Cloud Infrastructure (OCI), or Azure store using centralized configuration providers. This store contains connection information, such as connect descriptors and tuning parameters. Secrets, such as passwords and wallets, can be stored in OCI Vault or Azure Key Vault with the same providers.

Centralized application configuration makes client configuration and secrets management across multiple deployments simpler, especially for modern architectures, such as microservices, cloud application services, and serverless apps.

See Also

[Centralized Configuration Providers for Deployments](#)

- **Managed ODP.NET and ODP.NET Core - Azure Active Directory Single Sign-On**

ODP.NET can now request the Microsoft Entra ID (Azure AD) OAuth2 access token directly from the Entra ID endpoint in addition to the prior methods of receiving the token through a file location or the .ODP.NET API. This simplifies the use of the OAuth2 tokens since applications don't need to be modified or a separate helper utility isn't needed to get the token.

Users can sign-on once with Microsoft Entra ID, acquire the token, and access their on-premises and cloud-based Oracle AI Databases. This feature is available in ODP.NET Core and managed ODP.NET. This multicloud capability eases authentication and authorization between Microsoft Entra ID and Oracle AI Databases by simplifying user access and management.

See Also

[Using Microsoft Entra ID](#)

- **Managed ODP.NET and ODP.NET Core – Oracle IAM SSO token**

ODP.NET can now request the OCI IAM access token directly from IAM in addition to the prior methods of receiving the token through a file location or the .ODP.NET API. This simplifies the use of the IAM tokens since applications do not need to be modified or a separate helper utility is not needed to get the token.

Users can sign-on once with OCI IAM, acquire the token, and access their OCI DBaaS Oracle AI Databases. This feature is available in ODP.NET Core and managed ODP.NET.

① See Also

[Connecting to Oracle Database](#)

- Managed ODP.NET and ODP.NET Core - TLS/SSL Certificate Selection

Managed ODP.NET and ODP.NET Core allow selecting a specific TLS/SSL certificate stored in the Microsoft Certificate Store wallet location via a graphical user interface or thumbprint. The selected key is cached so that it doesn't have to be re-selected on subsequent connection attempts.

The graphical interface selection is enabled by setting the `OracleConnection.AllowCertificateSelectionUI` property to true. Thumbprint selection is enabled by populating the `SSLCertificateThumbprint` property with the thumbprint value.

This feature was backported to the latest ODP.NET 19c and 21c release updates.

- Managed ODP.NET and ODP.NET Core – Enhanced Transaction Management Options

By default, ODP.NET auto-commits after every SQL operation execution outside of a transaction. Managed ODP.NET and ODP.NET Core now allow disabling auto-commit so that SQL statements can be executed after an explicit commit without requiring a transaction.

The `OracleConnection` class adds new APIs to manage local transactions.

These new capabilities provide ODP.NET developers flexible and simpler transaction management options.

① See Also

[Command Auto-Commit](#)

- Data Use Case Domain Name and Schema

A Data Use Case Domain is a dictionary object that belongs to a schema and encapsulates a set of optional properties and constraints for common values, such as credit card numbers or email addresses. After you define a Data Use Case Domain, you can define table columns to be associated with that domain, thereby explicitly applying the domain's optional properties and constraints to those columns.

With Data Use Case Domains, you can define how you intend to use data centrally. They make it easier to ensure you handle values consistently across applications and improve data quality.

All ODP.NET provider types support using Data Use Case Domains, as well as retrieving Data Use Case Domain schema and name information

① See Also

[Data Use Case Domains](#)

- Annotations

Oracle AI Database annotations enable storing and retrieving metadata about database objects. These are either name-value pairs or only a name. These are free-form text fields

that applications can use to customize business logic or user interfaces. All ODP.NET provider types support Oracle AI Database annotations.

Annotations help you to use database objects in the same way, across all applications. This simplifies development and improves data quality.

See Also

[Annotations](#)

- Increased Database Password Length

Starting with this release, Oracle AI Database and client drivers including ODP.NET managed, unmanaged, and core drivers support passwords up to 1024 bytes in length.

- Transport Layer Security 1.3

ODP.NET supports Transport Layer version 1.3 for the core, managed, and unmanaged providers. TLS is used to encrypt data and authenticate connections. TLS 1.3 offers better security and performance over TLS 1.2.

See Also

[Using Transport Layer Security and Secure Sockets Layer](#)

- Managed ODP.NET and ODP.NET Core - Programmatic Database Startup and Shutdown Operations

Users with database administrator privileges can use the `OracleDatabase` class to startup or shutdown a database instance with managed ODP.NET and ODP.NET Core.

See Also

[OracleDatabase Class](#)

- Allowed Logon Version Client Support

This feature enables applications to specify the minimum authentication protocol that is to be used for a given application and/or a given `OracleConnection` object when authenticating against an Oracle AI Database.

See Also

[Allowed Logon Version Client Support](#)

- Managed ODP.NET - Public `OracleMigrationSqlGenerator` class

For Entity Framework 6 apps, the `OracleMigrationSqlGenerator` class is now public to allow developers to override the default SQL generation if they want to customize it.

① See Also

[Unsupported Entity Framework Features](#)

- HTTPS Proxy for Cloud Connections

ODP.NET supports HTTPS proxy of the database connection. The feature enables tunneling of that connection over a forward HTTP proxy using the HTTP CONNECT method.

It can be enabled in the connect address, such as

```
(description=(address=(https_proxy=www-proxy.example.com)(https_proxy_port=80)
(protocol=tcps)(port=1522)(host=dbhost.example.com))
(connect_data=(service_name=svc.example.com)) )
```

HTTPS proxy improves public cloud database service access as it eliminates requiring the client side firewall to open an outbound port.

Deprecated Features

The following features are deprecated in Oracle Data Provider for .NET for Release 26ai:

- Oracle Data Provider for .NET, Unmanaged Driver

Oracle Data Provider for .NET (ODP.NET), Unmanaged Driver is deprecated in Oracle AI Database 26ai, and can be desupported in a future release. Oracle recommends existing unmanaged ODP.NET applications migrate to ODP.NET, Managed Driver.

ODP.NET provides ADO.NET-based data access to Oracle databases. There are two primary Oracle data access drivers for Microsoft .NET Framework: ODP.NET, Managed Driver and ODP.NET, Unmanaged Driver.

In Oracle AI Database 26ai, ODP.NET, Managed Driver supports all major features available in ODP.NET, Unmanaged Driver with the same application programming interfaces and configuration settings. Code migration from unmanaged ODP.NET to managed ODP.NET is straightforward for the vast majority of existing .NET applications. ODP.NET, Managed Driver is a more compact and simpler install that can be consumed via NuGet packaging. Unmanaged ODP.NET is not available as a NuGet package. It is easier to manage multiple managed ODP.NET deployments on the same machine than multiple unmanaged ODP.NET deployments. These advantages make managed ODP.NET preferable for customer use over unmanaged ODP.NET.

As unmanaged ODP.NET no longer has any advantages over managed ODP.NET, Oracle has chosen to deprecate unmanaged ODP.NET.

- The mkstore wallet management command line tool

The mkstore command-line utility is being deprecated in favor of orakpi.

- MY_WALLET_DIRECTORY

① See Also

- *Oracle AI Database Upgrade Guide* for a complete list of deprecated features.

Desupported Features

Some features previously described in this document are desupported in Oracle AI Database 26ai. See *Oracle AI Database Upgrade Guide* for a complete list of desupported features.

The following features are no longer supported by Oracle:

- Oracle Database Extensions for .NET

Oracle Database Extensions for .NET is desupported. Oracle recommends that you either place .NET code in the middle tier, or use the External Procedures feature, or rewrite the code using PL/SQL or Java.

Oracle Database Extensions for .NET is a feature of Oracle Database on Microsoft Windows that enables you to use stored procedures and functions written in a language managed by .NET, such as C#.

Oracle Database hosts the Microsoft Common Language Runtime (CLR) in an external process, outside of the Oracle Database process. Application developers can write stored procedures and functions using any .NET compliant language, such as C# and VB.NET, and use these .NET stored procedures in the database, in the same manner as other PL/SQL or Java stored procedures. .NET stored procedures can be called from PL/SQL packages, procedures, functions, and triggers; from SQL statements; or from anywhere a PL/SQL procedure or function can be called.

Migration options include:

- Moving the .NET code (assemblies) into a middle tier
- Using the External Procedures feature to have the external process load and execute the .NET assembly
- Rewriting the stored procedures using PL/SQL or Java
- Oracle Wallet Manager (OWM)
- OracleConfiguration.DirectoryType Property

Changes in Oracle Data Provider for .NET Release 21c (21.12)

The following are the changes in Oracle Data Provider for .NET for Release 21c (21.12).

New Features

The following features are new in this release:

- .NET 8 Runtime

ODP.NET Core is certified for .NET 8.

- Entity Framework Core 8

ODP.NET Entity Framework (EF) Core now supports Entity Framework Core 8 with Oracle databases. ODP.NET EF Core 21.12.1 or higher is required.

Changes in Oracle Data Provider for .NET Release 21c (21.8)

The following are the changes in Oracle Data Provider for .NET for Release 21c (21.8).

New Features

The following features are new in this release:

- .NET 7 Runtime

ODP.NET Core is certified for .NET 7.

- Entity Framework Core 7

ODP.NET Entity Framework (EF) Core now supports Entity Framework Core 7 with Oracle databases. Entity Framework Core is a cross-platform Microsoft object-relational mapper that enables .NET developers to work with relational databases using .NET objects.

Changes in Oracle Data Provider for .NET Release 21c (21.7)

The following are the changes in Oracle Data Provider for .NET for Release 21c (21.7).

New Features

The following feature is new in this release:

- Managed ODP.NET and ODP.NET Core -- Microsoft Entra ID

Managed ODP.NET and ODP.NET Core 21.7 support Microsoft Entra ID, earlier known as Azure Active Directory authentication when connecting to Oracle Database. ODP.NET will then use an access token to authenticate instead of an username and password.

This feature benefits applications and services that use Microsoft Entra ID for centralized user authentication with Oracle database. Those services can include Azure and Microsoft 365-based cloud services, such as Microsoft Power BI service, that rely on Microsoft Entra ID for user management.

Using token-based authentication is more secure and simpler for the end user. It becomes unnecessary to specify credentials each time the user accesses a resource. Moreover, the resource never needs to handle and manage individual user credentials.

① See Also

[Using Microsoft Entra ID](#)

- Managed ODP.NET and ODP.NET Core -- Transport Layer Security 1.3

Managed ODP.NET and ODP.NET Core have added support for version 1.3 of Transport Layer Security (TLS) protocol. TLS allows applications to communicate securely with the database in a way that prevents message eavesdropping, tampering, and forgery.

① See Also

[Using Transport Layer Security and Secure Sockets Layer](#)

Changes in Oracle Data Provider for .NET Release 21c (21.6.1)

The following are the changes in Oracle Data Provider for .NET for Release 21c (21.6.1).

New Features

The following feature is new in this release:

- Managed ODP.NET and ODP.NET Core - Oracle Identity and Access Management Cloud Service Enhancements

ODP.NET adds more support for Oracle Identity and Access Management (IAM) cloud service for unified identity across Oracle cloud services for the managed and core drivers. ODP.NET can use the same Oracle IAM credentials for authentication and authorization.

This capability allows single sign-on and for identity to be propagated to all services Oracle IAM supports. A unified identity makes user management and account management easier for administrators and end users.

Managed ODP.NET adds support for IAM database password verifier, token, and database alternate password for token authentication.

ODP.NET Core already has support for IAM database password verifier and token. This new version adds database alternate password for token authentication in ODP.NET Core.

① See Also

[Connecting to Oracle Autonomous Database](#)

- ODP.NET Core - Multi-Platform Secure External Password Store

The Secure External Password Store (SEPS) is the use of a client-side wallet for securely storing the password credentials. ODP.NET Core can now be configured to use the external password store on any operating system it supports beyond just Windows.

An Oracle wallet is a container that securely stores authentication and signing credentials. Wallets can simplify large-scale deployments that rely on password credentials for database connections. Applications no longer need embedded user names and passwords, which reduces security risk.

① See Also

[Using Secure External Password Store](#)

Changes in Oracle Data Provider for .NET Release 21c (21.5)

The following are the changes in Oracle Data Provider for .NET for Release 21c (21.5).

New Features

The following feature is new in this release:

- Unmanaged ODP.NET – One-way TLS/SSL with Built-in Truspoints (Walletless)

Unmanaged ODP.NET now supports one-way Transport Layer Security/Secure Sockets Layer without wallets. Not having to provide a wallet can simplify database connectivity, such as with Oracle Autonomous Database.

Changes in Oracle Data Provider for .NET Release 21c (21.4.1)

The following are the changes in Oracle Data Provider for .NET for Release 21c (21.4.1).

New Features

The following features are new in this release:

- .NET 6 Runtime Certification
ODP.NET Core is certified for .NET 6.
- Entity Framework Core 6
ODP.NET Entity Framework (EF) Core now supports Entity Framework Core 6 with Oracle databases. Entity Framework Core is a cross-platform Microsoft object-relational mapper that enables .NET developers to work with relational databases using .NET objects.
- ODP.NET Core - Oracle Identity and Access Management Cloud Service
ODP.NET supports Oracle Identity and Access Management (IAM) cloud service for unified identity across Oracle cloud services, including Oracle Cloud Database Services, with the core driver. ODP.NET can use the same Oracle IAM credentials for authentication and authorization.

This capability allows single sign-on and for identity to be propagated to all services Oracle IAM supports. A unified identity makes user management and account management easier for administrators and end users.

① See Also

[Connecting to Oracle Autonomous Database](#)

Changes in Oracle Data Provider for .NET Release 21c (21.4)

The following are the changes in Oracle Data Provider for .NET for Release 21c (21.4).

New Features

The following features are new in this release:

- Managed ODP.NET and ODP.NET Core – One-way TLS/SSL with Built-in Truspoints (Walletless)
ODP.NET core and managed now support one-way Transport Layer Security/Secure Sockets Layer without wallets. Not having to provide a wallet can simplify database connectivity, such as with Oracle Autonomous Database.
- ODP.NET Core -- Ubuntu and Debian Linux
ODP.NET Core add certification for additional Linux distributions: Ubuntu and Debian.

① See Also[System Requirements](#)

Changes in Oracle Data Provider for .NET Release 21c (21.3)

The following are the changes in Oracle Data Provider for .NET for Release 21c (21.3).

New Features

The following feature is new in this release:

- Managed ODP.NET and ODP.NET Core – User-Defined Types

ODP.NET Core and managed providers now support user-defined types (UDT), including Oracle Collections (VARRAY and nested tables), Oracle Objects, and references (REF) to object types. These new features provide near-parity functionality with existing ODP.NET, Unmanaged Driver UDT functionality. This new enhancement allows the vast majority of unmanaged ODP.NET applications to migrate to managed ODP.NET or ODP.NET Core easily with minimal code changes necessary.

① See Also[Oracle User-Defined Types \(UDTs\) and .NET Custom Types](#)

- OracleConfiguration SqlnetURI Setting

ODP.NET OracleConfiguration SqlnetURI property is being renamed to SqlNetURI. This setting specifies the WebSocket universal resource identifier. The change reflects improved adherence to .NET's PascalCasing capitalization convention. If you currently use the property, then please modify your code accordingly.

Deprecated Features

The following property and setting are deprecated in Oracle Data Provider for .NET for Release 21c:

- OracleConfiguration DirectoryType property
- Configuration file DIRECTORY_TYPE setting

The OracleConfiguration DirectoryServerType property replaces the DirectoryType property. The .NET configuration file DIRECTORY_SERVER_TYPE setting replaces the DIRECTORY_TYPE setting. All these properties have identical functionality. Oracle recommends developers to use and migrate to the new properties. The DirectoryServerType and DIRECTORY_SERVER_TYPE names better align with the ldap.ora parameter DIRECTORY_SERVER_TYPE, which provides equivalent functionality.

- [Deprecation of Oracle Database Extensions for .NET](#)

① See Also

- *Oracle Database Upgrade Guide* for a complete list of deprecated features.

Deprecation of Oracle Database Extensions for .NET

Oracle Database Extensions for .NET is deprecated in Oracle Database 21c. Oracle recommends that you either place .NET code in the middle tier, or use the External Procedures feature, or rewrite the code using PL/SQL or Java.

Date: September 2021

Oracle Database Extensions for .NET is a feature of Oracle Database on Microsoft Windows that enables you to use stored procedures and functions written in a language managed by .NET, such as C#.

Oracle Database hosts the Microsoft Common Language Runtime (CLR) in an external process, outside of the Oracle Database process. Application developers can write stored procedures and functions using any .NET compliant language, such as C# and VB.NET, and use these .NET stored procedures in the database, in the same manner as other PL/SQL or Java stored procedures. .NET stored procedures can be called from PL/SQL packages, procedures, functions, and triggers; from SQL statements; or from anywhere a PL/SQL procedure or function can be called.

Migration options include:

- Moving the .NET code (assemblies) into a middle tier
- Using the External Procedures feature to have the external process load and execute the .NET assembly
- Rewriting the stored procedures using PL/SQL or Java

Changes in Oracle Data Provider for .NET (21.1)

The following are the changes in Oracle Data Provider for .NET (21.1).

New Features

The following features are new in this release:

- Entity Framework Core 5

ODP.NET Entity Framework (EF) Core now supports Entity Framework Core 5 with Oracle databases. Entity Framework Core is a cross-platform Microsoft object-relational mapper that enables .NET developers to work with relational databases using .NET objects.

① See Also

[Oracle Data Provider for .NET Entity Framework Core](#)

- Oracle Database JSON Data Type

ODP.NET supports the native JavaScript Object Notation (JSON) data type in Oracle Database. The new JSON data type is optimized for query and DML processing, yielding database performance improvements processing JSON.

When using a .NET string or `OracleString`, JSON data can be bound as a parameter using the `OracleDbType.Json` enumeration value. ODP.NET Core, managed, and unmanaged all support these new JSON features.

- Client Initiated Continuous Query Notification

Client Initiated Continuous Query Notification (CICQN) is similar to the traditional Continuous Query Notification (CQN) feature available in ODP.NET. In CQN, applications receive client-side notifications when server side change occurs that would affect the client's query result set, the underlying schema objects, or the database state. This notification is out of process, occurring without an existing ODP.NET connection.

CICQN uses in-process notifications. ODP.NET creates one separate connection per pool for receiving notifications. The database server uses this specific connection to send change notifications to ODP.NET. CICQN is useful when out of process communications is not available between client and database server, such as in cloud deployments.

 See Also

[Client Initiated Continuous Query Notifications](#)

- Managed ODP.NET and ODP.NET Core – Oracle Globally Distributed Database

ODP.NET Core and managed providers now support sharding. Oracle Globally Distributed Database provides the ability to horizontally partition the data across multiple independent Oracle databases (shards). Based on a key specified in the connect string, ODP.NET can route the database requests to a particular shard. Oracle Globally Distributed Database is a shared-nothing architecture that allows near-linear scaling of the database across low-cost commodity database servers located in one or more local or global data centers. Other key benefits include global data distribution (store particular data close to consumers) and fault containment (failure of one shard does not affect the availability of other shards). Global Data Services manages the location of data among the shards and allows ODP.NET client requests to be routed to the appropriate shard in this distributed database system.

In addition to the same sharding functionality that unmanaged ODP.NET supports, ODP.NET managed and core also support pausing connection requests during chunk migrations. Users will not experience a timeout without giving the chunk migration sufficient times to move across shards.

 See Also

[Oracle Globally Distributed Database](#)

- New Administrative Privileges

ODP.NET now supports assignment of task-specific and least-privileged administrative privileges to enable database administrative duty separation. The newly added privileges include SYSBACKUP for backup and recovery, SYSDG for Oracle Data Guard, SYSKM for encryption key management, and SYSRAC for Oracle Real Applications Clusters operations. Core, managed, and unmanaged ODP.NET support all these administrative privileges.

① See Also

[OracleDBAPrivilege Enumeration](#)

- Managed ODP.NET and ODP.NET Core-- Debug Tracing Redaction

Managed ODP.NET and ODP.NET Core has introduced a new trace level, that can exclude SQL statements and network packet contents from being included in the trace file.

① See Also

[Debug Tracing](#)

Changes in Oracle Data Provider for .NET (19.16)

The following are the changes in Oracle Data Provider for .NET (19.16).

New Features

The following features are new in this release:

- Managed ODP.NET and ODP.NET Core -- Transport Layer Security 1.3

Managed ODP.NET and ODP.NET Core have added support for version 1.3 of Transport Layer Security (TLS) protocol. TLS allows applications to communicate securely with the database in a way that prevents message eavesdropping, tampering, and forgery.

① See Also

[Using Transport Layer Security and Secure Sockets Layer](#)

Changes in Oracle Data Provider for .NET (19.15.1)

The following are the changes in Oracle Data Provider for .NET (19.15.1).

New Features

The following feature is new in this release:

- Microsoft Entra ID

Starting with ODP.NET 21.7 and 19.15.1, ODP.NET supports Microsoft Entra ID, earlier known as Azure Active Directory authentication when connecting to Oracle Database. ODP.NET will then use an access token to authenticate instead of a username and password.

This feature benefits applications and services that use Microsoft Entra ID for centralized user authentication with Oracle database. Those services can include Azure and Microsoft 365-based cloud services, such as Microsoft Power BI service, that rely on Microsoft Entra ID for user management.

Using token-based authentication is more secure and simpler for the end user. It becomes unnecessary to specify credentials each time the user accesses a resource. Moreover, the resource never needs to handle and manage individual user credentials.

See Also

[Using Microsoft Entra ID.](#)

Changes in Oracle Data Provider for .NET (19.10)

The following are the changes in Oracle Data Provider for .NET (19.10).

New Features

The following features are new in this release:

- .NET 5 Runtime Certification

ODP.NET Core is certified for .NET 5.

- Managed ODP.NET and ODP.NET Core -- Bulk Copy

ODP.NET Bulk Copy enables applications to efficiently load large amounts of data from a table in one database to another table in a different database. Managed ODP.NET and ODP.NET Core now support Bulk Copy and all its APIs.

ODP.NET Bulk Copy is the most optimized .NET solution when a large data set needs to be loaded into a table or between database tables in different databases.

See Also

[Bulk Copy](#)

- Oracle Provider Types Deserialization into DataSet and DataTable

Due to a change in all .NET versions to enhance application security, the allowed DataSet and DataTable data types that can be deserialized are now restricted. This change applies to .NET 5, .NET Core, and .NET Framework with new updates. If application DataSets and DataTables use ODP.NET data types with one of these newer .NET versions, then they may encounter an ODP.NET type initializer exception. To avoid this exception, developers can add ODP.NET-specific data types to the “allow” list to permit deserialization into DataSet or DataTable.

To do this, call the `OracleConfiguration.AddOracleTypesDeserialization` method or add the types individually via a .NET configuration file.

See Also

[Deserializing ODP.NET Types into DataSet and DataTable](#)

- ODP.NET Core -- KeepAlive on non-Windows platforms

KeepAlive is supported on non-Windows operating systems, such as Oracle Linux. In previous releases, ODP.NET Core supported these KeepAlive properties for Windows platforms only. The properties include KeepAlive, KeepAliveInterval, and KeepAliveTime.

KeepAlive is used to prevent idle TCP connections from being closed, such as by a firewall or load balancer. In some cloud deployments, this idle TCP connection timeout cannot be changed. KeepAlive will keep the connection alive by periodically sending a probe packet with no data in it and the ACK flag turned on.

See Also

[OracleConnection Properties](#)

- Suppress GetDecimal Invalid Cast Exception

The SuppressGetDecimalInvalidCastException property has been added to the OracleDataReader and OracleDataAdapter classes. When enabled and GetDecimal is called on the OracleDataReader object explicitly by the application or implicitly through the Fill() method on the OracleDataAdapter object, for example, it suppresses the InvalidCastException and returns a rounded-off 28 or 29 precision .NET decimal value, that represents the Oracle NUMBER.

See Also

[SuppressGetDecimalInvalidCastException](#)

- ODP.NET Core -- LDAP for Non-Windows Platforms

ODP.NET Core now supports connecting with net service names mapped to connect descriptors in an LDAP-compliant directory server from all non-Windows operating systems the provider supports, such as Oracle Linux and Red Hat Enterprise Linux. Now ODP.NET Core apps on all platforms can use LDAP authentication to connect to Oracle Database.

Changes in Oracle Data Provider for .NET (19.9)

The following are the changes in Oracle Data Provider for .NET (19.9).

New Features

The following feature is new in this release:

- In-Band Fast Application Notification

In some scenarios, such as cloud deployments and when firewalls block notification messages between the database and client, out of band messages and using Oracle Notification Service may not be possible. ODP.NET can use in-band FAN notifications instead in these scenarios. In-band notifications rely on existing ODP.NET connections to communicate messages. ODP.NET will check for notifications every time a connection makes a database round trip, is checked in, or checked out. When a DOWN notification is received, ODP.NET will scan the pool for affected connections and close them.

This feature was introduced with managed ODP.NET and ODP.NET Core in 19.9. It was introduced in unmanaged ODP.NET in 19.10.

See Also

[In-Band Notification](#)

Changes in Oracle Data Provider for .NET ODAC Release 19c (19.3.2)

The following are the changes in Oracle Data Provider for .NET for ODAC Release 19c (19.3.2).

New Features

The following features are new in this release:

- Entity Framework Core 3.1

ODP.NET Entity Framework (EF) Core now supports Entity Framework Core 3.1 with Oracle databases. Entity Framework Core is a cross-platform Microsoft object-relational mapper that enables .NET developers to work with relational databases using .NET objects. This ODP.NET EF Core release adds support for relational views and materialized views. These views can be read-only or updatable.

Oracle EF Core 3.1 developers can now create and migrate data models in the Oracle Database. They can scaffold EF Core classes based on an Oracle Database schema objects. They can migrate and scaffold with Oracle views. Oracle EF Core is integrated with popular EF Core tools, such as the EF Core Package Manager Tools.

See Also

[Oracle Data Provider for .NET Entity Framework Core](#)

Changes in Oracle Data Provider for .NET (19.7)

The following are the changes in Oracle Data Provider for .NET (19.7).

New Features

The following features are new in this release:

- Managed ODP.NET and ODP.NET Core -- Administer Storage using SYSASM Privilege

Oracle Automatic Storage Management (Oracle ASM) is a volume manager and a file system for Oracle database files. SYSASM is a system privilege that enables administrators to manage ASM instances. Managed ODP.NET and ODP.NET Core can now connect using the SYSASM administrative privilege to perform storage management of the Oracle Database.

ODP.NET can now perform database ASM administration when connecting with a SYSASM privileged connection.

See Also

[OracleDBAPrivilege Enumeration](#)

- WebSocket and WebSocket with SSL/TLS

WebSocket is a protocol that offers full-duplex communication channels over a single TCP connection. WebSocket with SSL/TLS offers a secure WebSocket connection. WebSocket is an extension to HTTP and is able to work with HTTP proxies and intermediaries. ODP.NET Core, managed, and unmanaged providers all support WebSocket and secure WebSocket protocols.

See Also

[Using WebSocket](#)

Changes in Oracle Data Provider for .NET (19.6)

The following are the changes in Oracle Data Provider for .NET (19.6).

New Features

The following features are new in this release:

- .NET Core 3.1 Certification
ODP.NET Core is certified for .NET Core 3.1.
- Linux 8 Certification
ODP.NET Core is certified on Oracle Linux 8 and Red Hat Enterprise Linux 8.
- Entity Framework 6.4 Certification
Managed and unmanaged ODP.NET Entity Framework is certified for Entity Framework 6.4.

Changes in Oracle Data Provider for .NET (19.5)

The following are the changes in Oracle Data Provider for .NET (19.5).

New Features

The following features are new in this release:

- .NET Core 3 Certification
ODP.NET Core is certified for .NET Core 3.
- Entity Framework Core 2.1 Certification
ODP.NET Entity Framework Core is certified for EF Core 2.1.

Changes in Oracle Data Provider for .NET in ODAC Release 19c (19.3)

The following are the changes in Oracle Data Provider for .NET for ODAC Release 19c (19.3).

New Features

The following features are new in this release:

- Entity Framework Core

ODP.NET Entity Framework (EF) Core is a database provider that allows Entity Framework Core to be used with Oracle databases. Entity Framework Core is a cross-platform Microsoft object-relational mapper that enables .NET developers to work with relational databases using .NET objects.

Oracle EF Core developers can now create and migrate data models in the Oracle Database. They can reverse engineer/scaffold EF Core classes based on an Oracle Database schema. Oracle EF Core is integrated with popular EF Core tools, such as the EF Core Package Manager Tools.

① See Also

[Oracle Data Provider for .NET Entity Framework Core](#)

- Managed ODP.NET and ODP.NET Core -- More Configuration Options with New OracleConnection Properties

OracleConnection class introduces additional properties to configure ODP.NET connections. The new properties are:

- `KeepAlive`, `KeepAliveInterval`, and `KeepAliveTime` - specifies whether and the conditions under which to keep idle connections alive
- `TnsAdmin` - specifies the `tnsnames.ora` and `sqlnet.ora` directory
- `WalletLocation` - specifies the wallet directory location

① Note

[OracleConnection Class](#)

- Unmanaged ODP.NET -- Administer Storage using SYSASM Privilege

Oracle Automatic Storage Management (Oracle ASM) is a volume manager and a file system for Oracle database files. SYSASM is a system privilege that enables administrators to manage ASM instances. Unmanaged ODP.NET can now connect using the SYSASM administrative privilege to perform storage management of the Oracle Database.

ODP.NET can now perform database ASM administration when connecting with a SYSASM privileged connection.

Note

[OracleDBAPrivilege Enumeration](#)

- Dynamically Enabled Tracing and Size Limits

To improve diagnostics ease of use and productivity, ODP.NET trace output can now be enabled and disabled at runtime using the `OracleConfiguration.TraceLevel` property. To keep trace files down to a manageable size, the `TraceFileMaxSize` property can be set.

Administrators can isolate tracing to only known times or events when problems occur. This feature helps keep trace files down to the minimum size needed to investigate issues. If a trace file does become large, ODP.NET will write to a new trace file once the maximum file limit that is set in `TraceFileMaxSize` is reached. These features are available with core, managed, and unmanaged providers.

See Also

- [Debug Tracing](#)
- [OracleConfiguration Diagnostics and Tracing Properties](#)

Changes in Oracle Data Provider for .NET Release 19c (19.3)

The following are the changes in Oracle Data Provider for .NET for Release 19c (19.3).

New Features

The following features are new in this release:

- .NET Framework 4.8 Certification
ODP.NET Core, Managed, and Unmanaged Drivers are certified with .NET Framework 4.8.
- ODP.NET, Managed Driver -- Configuration as Code
Managed ODP.NET now supports `OracleConfiguration`, `OracleDataSourceCollection`, and `OracleOnsServerCollection` classes.
These classes allow developers to configure managed ODP.NET within source code in lieu of .NET configuration and Oracle configuration files.
- ODP.NET Core -- Lightweight Directory Access Protocol (LDAP)
ODP.NET Core can use connect identifiers mapped to connect descriptors in an LDAP-compliant directory server, such as Oracle Internet Directory and Microsoft Active Directory. The provider supports the same LDAP features and settings as managed ODP.NET.
ODP.NET Core LDAP support is available on Windows operating systems only.

See Also

[OracleConfiguration Directories Properties](#)

- ODP.NET Core -- Performance Counters

ODP.NET Core can publish key runtime connection counters to Windows Performance Monitor or a file. The provider supports the same counters, setup process, and settings as managed and unmanaged ODP.NET.

ODP.NET Core performance counters are available on Windows operating systems only. They require System.Diagnostics namespace in using performance counters programmatically.

See Also

[Connection Performance Counters](#)

- Easy Connect Plus

Oracle Easy Connect Plus provides a simple way to configure TCP/IP connections to the Oracle Database without having to use parameter files, such as `tnsnames.ora`, nor environment variables. Easy Connect Plus includes support for:

- TCP/IP with SSL/TLS
- Any SQL*Net description level parameter can be used
- Multiple hosts and ports
- A straightforward name-value pair format

Easy Connect Plus supports more configurations and a wider breadth of ODP.NET applications than traditional Easy Connect, including clustered or cloud databases.

See Also

[Easy Connect and Easy Connect Plus Naming Methods](#)

- More Secure Oracle Notification Service Connections

ODP.NET now enables Oracle Notification Service (ONS) communications to occur over TCP/IP with SSL/TLS (TCPS), which is more secure than just TCP/IP. As TCPS requires using a wallet for storing keys and certificates, ODP.NET can use one wallet for both ONS and ODP.NET connections or have separate wallets for each.

TCPS provides more secure ONS communication, gives administrators flexibility in how to configure their wallets, and enables cloud database connections.

Introducing Oracle Data Provider for .NET

This chapter introduces Oracle Data Provider for .NET (ODP.NET), an implementation of a .NET data provider for Oracle Database.

This chapter contains these topics:

- [.NET Data Access in Oracle: Products and Documentation](#)
- [Overview of Oracle Data Provider for .NET \(ODP.NET\)](#)
- [Oracle Data Provider for .NET Assemblies](#)
- [Differences between the ODP.NET Drivers](#)
- [Getting Started With Developing ODP.NET Applications](#)

.NET Data Access in Oracle: Products and Documentation

This section discusses Oracle Data Provider for .NET and Oracle Database components that use Oracle Data Provider for .NET for data access. It briefly describes what each component does and where to find additional documentation.

These Oracle products provide .NET integration on the Windows operating system:

Oracle Data Provider for .NET (ODP.NET)

Oracle Data Provider for .NET provides fast data access from .NET clients to Oracle databases. ODP.NET enables .NET applications to take advantage of Oracle advanced features, such as Oracle Real Application Clusters (Oracle RAC) and XML DB. It is accessible through any .NET language, including C#, Visual Basic .NET, and C++ .NET.

ODP.NET consists of three drivers: ODP.NET, Managed Driver, ODP.NET, Unmanaged and ODP.NET Core. ODP.NET, Managed Driver is a fully managed ADO.NET provider, consisting of fewer DLLs and smaller install size than ODP.NET, Unmanaged Driver. The managed driver has the same exact application programming interfaces (APIs) as ODP.NET, Unmanaged Driver. However, the managed driver's APIs are a subset of the Unmanaged Driver's APIs.

ODP.NET Core is a multi-platform provider for Microsoft .NET Core. In functionality, it is very similar to ODP.NET, Managed Driver. However, ODP.NET Core has a subset of managed ODP.NET's APIs.

This guide describes Oracle Data Provider for .NET features, their use, installation, requirements, and classes. The guide distinguishes which classes and APIs are supported for the managed driver, unmanaged driver, and .NET clients.

Additionally, Oracle Data Provider for .NET Dynamic Help, which is context-sensitive online help, contains the same reference sections available in *Oracle Data Provider for .NET Developer's Guide for Microsoft Windows*, this guide.

Oracle Data Provider for .NET Dynamic Help is integrated with Visual Studio Dynamic Help. With Dynamic Help, you can access Oracle Data Provider for .NET documentation within Visual Studio by placing the cursor on an Oracle Data Provider for .NET keyword and pressing the F1 function key.

Oracle Developer Tools for Visual Studio

Oracle Developer Tools is an add-in to Visual Studio that provides graphical user interface (GUI) access to Oracle functionality. It provides improved developer productivity and ease of use.

Oracle Developer Tools for Visual Studio Help describes Oracle Developer Tools. This help is in the form of dynamic help, which installs as part of the product.

Additionally, the Oracle Developer Tools for Visual Studio Help includes the following documentation:

- *Oracle AI Database PL/SQL Language Reference*
- *Oracle AI Database SQL Language Reference*
- *Oracle AI Database Error Messages Reference*
- Access to Oracle Data Provider for .NET Dynamic Help
- Access to Oracle Providers for ASP.NET Dynamic Help

Oracle Providers for ASP.NET

Oracle Providers for ASP.NET offer ASP.NET developers an easy to use method to store state common to web applications within an Oracle database. These providers are modeled on existing Microsoft ASP.NET providers, sharing similar schema and programming interfaces to provide .NET developers a familiar interface. Oracle supports the following providers:

- Cache Dependency Provider
- Membership Provider
- Profile Provider
- Role Provider
- Session State Provider
- Site Map Provider
- Web Events Provider
- Web Parts Personalization Provider

Oracle Providers for ASP.NET classes, their use, installation, and requirements are described in *Oracle Providers for ASP.NET Developer's Guide for Microsoft Windows*, which is also provided as dynamic help.

Oracle Services for Microsoft Transaction Server

Oracle Services for Microsoft Transaction Server (OraMTS) permit Oracle databases to be used as resource managers in Microsoft application coordinated transactions. OraMTS acts as a proxy for the Oracle database to the Microsoft Distributed Transaction Coordinator (MSDTC). As a result, OraMTS provides client-side connection pooling and allows client components that leverage Oracle to participate in promotable and distributed transactions. In addition, OraMTS can operate with Oracle databases running on any operating system, given that the services themselves are run on Windows.

① See Also

Oracle Services for Microsoft Transaction Server Developer's Guide for Microsoft Windows for description about OraMTS, which allows Oracle databases to be used as resource managers in distributed transactions.

Oracle TimesTen In-Memory Database

ODP.NET support for Oracle TimesTen In-Memory Database (TimesTen) provides fast and efficient ADO.NET data access for applications that require the highest performance.

You can use ODP.NET with any of the following TimesTen installations:

- TimesTen Data Manager only (for direct connections)
- TimesTen Client only (for client/server connections, assuming a TimesTen Data Manager instance and TimesTen Server instance are accessible elsewhere)
- TimesTen Data Manager with TimesTen Server

For more information on ODP.NET features specific to a TimesTen environment, refer to the *Oracle Data Provider for .NET Oracle TimesTen In-Memory Database Support User's Guide*.

① Note

TimesTen does not support ODP.NET, Managed Driver and ODP.NET Core.

Overview of Oracle Data Provider for .NET (ODP.NET)

Oracle Data Provider for .NET (ODP.NET) is an implementation of a .NET data provider for Oracle Database, using and inheriting from classes and interfaces available in the [Microsoft .NET Framework Class Library](#).

Following the .NET Framework, ODP.NET uses the ADO.NET model, which allows native providers to expose provider-specific features and data types. This is similar to Oracle Provider for OLE DB, where ADO (ActiveX Data Objects) provides an automation layer that exposes an easy programming model. ADO.NET provides a similar programming model, but without the automation layer, for better performance.

Oracle Data Provider for .NET uses Oracle native APIs to offer fast and reliable access to Oracle data and features from any .NET application. ODP.NET consists of three drivers: ODP.NET, Managed Driver, ODP.NET, Unmanaged Driver, and ODP.NET Core. ODP.NET, Managed Driver is a fully managed ADO.NET provider, consisting of fewer DLLs and smaller install size than ODP.NET, Unmanaged Driver. The managed driver has the same exact application programming interfaces (APIs) as ODP.NET, Unmanaged Driver. However, the managed driver's APIs are a subset of the Unmanaged Driver's APIs.

ODP.NET Core employs the same namespaces and application programming interfaces (APIs) as ODP.NET, Managed Driver. This parallel eases migration and developer learning curve from managed ODP.NET to ODP.NET Core. It does not support all managed ODP.NET functionality. ODP.NET Core supports a subset of managed ODP.NET APIs. These differences are listed later on in this documentation.

The ODP.NET classes described in this guide are contained in the `Oracle.DataAccess.dll` and `Oracle.ManagedDataAccess.dll` assembly.

- Client Applications: All ODP.NET classes are available for use in client applications.

As ODP.NET, Managed Driver does not support all classes and members in the ODP.NET, Unmanaged Driver, the unsupported managed driver classes and members will be labeled *Not Supported in ODP.NET, Managed Driver*.

ODP.NET Core does not support all classes and members in the ODP.NET, Managed Driver. The unsupported managed driver classes and members will be labeled *Not Supported in ODP.NET Core*.

See Also

- [Oracle Data Provider for .NET Assemblies](#) for class lists

Oracle Data Provider for .NET Assemblies

This section contains the following topics:

- [Oracle Data Provider for .NET, Unmanaged Driver Assemblies](#)
- [Oracle Data Provider for .NET, Managed Driver and ODP.NET Core Assemblies](#)
- [Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)
- [Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)

Oracle Data Provider for .NET, Unmanaged Driver Assemblies

The `Oracle.DataAccess.dll` [assembly](#) provides two namespaces:

- The `Oracle.DataAccess.Client` namespace contains ODP.NET classes and enumerations for the client-side provider.
- The `Oracle.DataAccess.Types` namespace contains the Oracle Data Provider for .NET data types (ODP.NET Types).

To use Code First or Entity Framework 6 or higher with ODP.NET, Unmanaged Driver, add `Oracle.DataAccess.EntityFramework.dll` as a project assembly reference. It contains the namespace `Oracle.DataAccess.EntityFramework`.

Oracle Data Provider for .NET, Managed Driver and ODP.NET Core Assemblies

The `Oracle.ManagedDataAccess.dll` [assembly](#) provides two namespaces:

- The `Oracle.ManagedDataAccess.Client` namespace contains ODP.NET classes and enumerations for the client-side provider.
- The `Oracle.ManagedDataAccess.Types` namespace contains the Oracle Data Provider for .NET data types (ODP.NET Types).

ODP.NET, Managed Driver contains additional assemblies. These assemblies are optional to install if not using the specific functionality.

Applications do not need to explicitly add these assemblies to their project. ODP.NET, Managed Driver will access these assemblies by default if installed.

The one exception is `Oracle.ManagedDataAccess.EntityFramework.dll`. That DLL must be explicitly added to a project for its functionality to be used.

- `Oracle.ManagedDataAccess.EntityFramework.dll` - Only required when using Code First or Entity Framework 6 or higher. It contains the `Oracle.ManagedDataAccess.EntityFramework` namespace.
- `Oracle.ManagedDataAccessIOP.dll` - Only required when using MIT Kerberos. The assembly makes calls to unmanaged assemblies. Applications do not need to explicitly add this assembly to their project as ODP.NET is already configured to access this assembly by default.

Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces

The `Oracle.DataAccess.Client` and `Oracle.ManagedDataAccess.Client` namespaces contains implementations of core ADO.NET classes and enumerations for ODP.NET, as well as ODP.NET specific classes.

The following tables list ODP.NET classes, enumerations, and types that are supported by the `Oracle.DataAccess.Client` and `Oracle.ManagedDataAccess.Client` namespaces. The tables indicate which of them are not supported by ODP.NET, Managed Driver and/or ODP.NET Core. All are supported by ODP.NET, Unmanaged Driver unless noted as not supported in the description field.

Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client

[Table 1-1](#) lists the `Oracle.DataAccess.Client` and `Oracle.ManagedDataAccess.Client` classes and delegates.

Table 1-1 Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client

Class or Delegate	Supported in the ODP.NET, Managed Driver	Supported in the ODP.NET Core	Description
OnChangeEventHandler Delegate	-	-	The <code>OnChangeEventHandler</code> event delegate represents the signature of the method that handles the notification.
OracleAQAgent Class	-	-	The <code>OracleAQAgent</code> class represents agents that may be senders or recipients of a message.
OracleAQDequeueOptions Class	-	-	An <code>OracleAQDequeueOptions</code> object represents the options available when dequeuing a message from an <code>OracleAQQueue</code> object.

Table 1-1 (Cont.) Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client

Class or Delegate	Supported in the ODP.NET, Managed Driver	Supported in the ODP.NET Core	Description
OracleAQEnqueueOptions Class	-	-	The <code>OracleAQEnqueueOptions</code> class represents the options available when enqueueing a message to an <code>OracleAQQueue</code> .
OracleAQMessage Class	-	-	An <code>OracleAQMessage</code> object represents a message to be enqueued and dequeued.
OracleAQMessageAvailableEventArgs Class	-	-	The <code>OracleAQMessageAvailableEventArgs</code> class provides event data for the <code>OracleAQQueue.MessageAvailable</code> event.
OracleAQMessageAvailableEventHandler Delegate	-	-	The <code>OracleAQMessageAvailableEventHandler</code> delegate represents the signature of the method that handles the <code>OracleAQQueue.MessageAvailable</code> event.
OracleAQQueue Class	-	-	An <code>OracleAQQueue</code> object represents a queue.
OracleBulkCopy Class	-	-	An <code>OracleBulkCopy</code> object efficiently bulk loads or copies data into an Oracle table from another data source.
OracleBulkCopyOptions Enumeration	-	-	The <code>OracleBulkCopyOptions</code> enumeration specifies the values that can be combined with an instance of the <code>OracleBulkCopy</code> class and used as options to determine its behavior and the behavior of the <code>WriteToServer</code> methods for that instance.
OracleBulkCopyColumnMapping Class	-	-	The <code>OracleBulkCopyColumnMapping</code> class defines the mapping between a column in the data source and a column in the destination database table.

Table 1-1 (Cont.) Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client

Class or Delegate	Supported in the ODP.NET, Managed Driver	Supported in the ODP.NET Core	Description
OracleBulkCopyColumnMappingCollection Class	-	-	The <code>OracleBulkCopyColumnMappingCollection</code> class represents a collection of <code>OracleBulkCopyColumnMapping</code> objects that are used to map columns in the data source to columns in a destination table.
OracleClientFactory Class	-	No	An <code>OracleClientFactory</code> object allows applications to instantiate ODP.NET classes in a generic way.
OracleCommand Class	-	-	An <code>OracleCommand</code> object represents a SQL command, a stored procedure or function, or a table name.
OracleCommandBuilder Class	-	-	An <code>OracleCommandBuilder</code> object provides automatic SQL generation for the <code>OracleDataAdapter</code> when the database is updated.
OracleConfiguration Class	-	-	An <code>OracleConfiguration</code> is a static class for setting ODP.NET Core configuration data using a single programming interface.
OracleConnection Class	-	-	An <code>OracleConnection</code> object represents a connection to Oracle Database.
OracleConnectionOpenEventArgs Class	-	-	An <code>OracleConnectionOpenEventArgs</code> object provides connection information for the <code>OracleConnection.Open()</code> method.
OracleConnectionOpenEventHandler Delegate	-	-	The <code>OracleConnectionOpenEventHandler</code> delegate represents the signature of the method that handles <code>OracleConnection.ConnectionOpen</code> event.
OracleConnectionStringBuilder Class	-	-	An <code>OracleConnectionStringBuilder</code> object allows applications to create or modify connection strings.

Table 1-1 (Cont.) Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client

Class or Delegate	Supported in the ODP.NET, Managed Driver	Supported in the ODP.NET Core	Description
OracleCredential Class	-	-	OracleCredential class provides a secure way to provide password while opening connection with Oracle Database using the ODP.NET driver
OracleDataAdapter Class	-	-	An OracleDataAdapter object represents a data provider object that communicates with the DataSet.
OracleDatabase Class	-	-	An OracleDatabase object represents an Oracle Database instance.
OracleDataReader Class	-	-	An OracleDataReader object represents a forward-only, read-only, in-memory result set.
OracleDataSource Class	No	-	An OracleDataSource object represents an Oracle database from which to open connections to or execute commands directly against.
OracleDataSourceCollection Class	-	-	An OracleDataSourceCollection object supports adding and deleting network service name (i.e. TNS) entries in the OracleDataSourceCollection.
OracleDataSourceEnumerator Class	-	-	An OracleDataSourceEnumerator object allows applications to generically obtain a collection of data sources to connect to.
OracleDependency Class	-	-	An OracleDependency class represents a dependency between an application and an Oracle database.
OracleError Class	-	-	The OracleError object represents an error reported by an Oracle database.
OracleErrorCollection Class	-	-	An OracleErrorCollection object represents a collection of OracleErrors.

Table 1-1 (Cont.) Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client

Class or Delegate	Supported in the ODP.NET, Managed Driver	Supported in the ODP.NET Core	Description
OracleException Class	-	-	The <code>OracleException</code> object represents an exception that is thrown when Oracle Data Provider for .NET encounters an error.
OracleFailoverEventArgs Class	-	-	The <code>OracleFailoverEventArgs</code> class provides event data for the <code>OracleConnection.Failover</code> event.
OracleFailoverEventHandler Delegate	-	-	The <code>OracleFailoverEventHandler</code> represents the signature of the method that handles the <code>OracleConnection.Failover</code> event.
OracleGlobalization Class	-	-	The <code>OracleGlobalization</code> class is used to obtain and set the Oracle globalization settings of the session, thread, and local computer (read-only).
OracleHAEEventArgs Class	-	-	The <code>OracleHAEEventArgs</code> class provides event data for the <code>OracleConnection.HAEvent</code> event.
OracleHAEEventHandler Delegate	-	-	The <code>OracleHAEEventHandler</code> delegate represents the signature of the method that handles the <code>OracleConnection.HAEvent</code> event.
OracleInfoMessageEventArgs Class	-	-	The <code>OracleInfoMessageEventArgs</code> object provides event data for the <code>OracleConnection.InfoMessage</code> event.
OracleInfoMessageEventHandler Delegate	-	-	The <code>OracleInfoMessageEventHandler</code> delegate represents the signature of the method that handles the <code>OracleConnection.InfoMessage</code> event.

Table 1-1 (Cont.) Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client

Class or Delegate	Supported in the ODP.NET, Managed Driver	Supported in the ODP.NET Core	Description
OracleLogicalTransaction Class	-	-	The OracleLogicalTransaction class provides detailed information about the logical transaction status.
OracleNotificationEventArgs Class	-	-	The OracleNotificationEventArgs class provides event data for a notification.
OracleNotificationRequest Class	-	-	An OracleNotificationRequest class represents a notification request to be subscribed in the database.
OracleOnsServerCollection Class	-	-	The OracleOnsServerCollection class supports adding to and deleting from a collection of logical servers with their corresponding list of nodes where the Oracle Notification Service (ONS) daemons are talking to their remote clients.
OracleParameter Class	-	-	An OracleParameter object represents a parameter for an OracleCommand.
OracleParameterCollection Class	-	-	An OracleParameterCollection object represents a collection of OracleParameters.
OraclePermission Class	-	No	An OraclePermission object enables ODP.NET to enforce imperative security and helps ensure that a user has a security level adequate for accessing data.
OraclePermissionAttribute Class	-	No	An OraclePermissionAttribute object enables ODP.NET to enforce declarative security and helps ensure that a user has a security level adequate for accessing data.
OracleRowsCopiedEventHandler Delegate	-	-	The OracleRowsCopiedEventHandler delegate represents the method that handles the OracleRowsCopied event of an OracleBulkCopy object.

Table 1-1 (Cont.) Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client

Class or Delegate	Supported in the ODP.NET, Managed Driver	Supported in the ODP.NET Core	Description
OracleRowsCopiedEventArgs Class	-	-	The <code>OracleRowsCopiedEventArgs</code> class represents the set of arguments passed as part of event data for the <code>OracleRowsCopied</code> event.
OracleRowUpdatedEventArgs Class	-	-	The <code>OracleRowUpdatedEventArgs</code> object provides event data for the <code>OracleDataAdapter.RowUpdated</code> event.
OracleRowUpdatedEventHandler Delegate	-	-	The <code>OracleRowUpdatedEventHandler</code> delegate represents the signature of the method that handles the <code>OracleDataAdapter.RowUpdated</code> event.
OracleRowUpdatingEventArgs Class	-	-	The <code>OracleRowUpdatingEventArgs</code> object provides event data for the <code>OracleDataAdapter.RowUpdating</code> event.
OracleRowUpdatingEventHandler Delegate	-	-	The <code>OracleRowUpdatingEventHandler</code> delegate represents the signature of the method that handles the <code>OracleDataAdapter.RowUpdating</code> event.
OracleShardingKey Class	-	-	An <code>OracleShardingKey</code> object can represent either a sharding key or a super sharding key.
OracleTAFMode Class	-	-	The <code>OracleTAFMode</code> object sets the Oracle Transparent Application Failover parameters for the connection. <i>Not supported with unmanaged ODP.NET.</i>
OracleTransaction Class	-	-	An <code>OracleTransaction</code> object represents a local transaction.

Table 1-1 (Cont.) Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client

Class or Delegate	Supported in the ODP.NET, Managed Driver	Supported in the ODP.NET Core	Description
OracleXmlQueryProperties Class	-	-	An <code>OracleXmlQueryProperties</code> object represents the XML properties used by the <code>OracleCommand</code> class when the <code>XmlCommandType</code> property is <code>Query</code> .
OracleXmlSaveProperties Class	-	-	An <code>OracleXmlSaveProperties</code> object represents the XML properties used by the <code>OracleCommand</code> class when the <code>XmlCommandType</code> property is <code>Insert</code> , <code>Update</code> , or <code>Delete</code> .

Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Enumerations

[Table 1-2](#) lists the client enumerations.

Table 1-2 Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Enumerations

Enumeration	Supported in the ODP.NET, Managed Driver and ODP.NET Core	Description
FailoverEvent Enumeration	No	<code>FailoverEvent</code> enumerated values are used to specify the state of the failover.
FailoverReturnCode Enumeration	No	<code>FailoverReturnCode</code> enumerated values are passed back by the application to the ODP.NET provider to request a retry in case of a failover error, or to continue in case of a successful failover.
FailoverType Enumeration	No	<code>FailoverType</code> enumerated values are used to indicate the type of failover event that was raised.
OracleAQDequeueMode Enumeration	-	The <code>OracleAQDequeueMode</code> enumeration type specifies the dequeue mode.
OracleAQMessageDeliveryMode Enumeration	-	The <code>OracleAQMessageDeliveryMode</code> enumeration type specifies the delivery mode of the message.
OracleAQMessageState Enumeration	-	The <code>OracleAQMessageState</code> enumeration type identifies the state of the message at the time of dequeue.
OracleAQMessageType Enumeration	-	The <code>OracleAQMessageType</code> enumeration type specifies the message payload type.

Table 1-2 (Cont.) Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Enumerations

Enumeration	Supported in the ODP.NET, Managed Driver and ODP.NET Core	Description
OracleAQNavigationMode Enumeration	-	The OracleAQNavigationMode enumeration type specifies the navigation mode.
OracleAQNotificationGroupingType Enumeration	-	The OracleAQNotificationGroupingType enumeration type specifies the notification grouping type.
OracleAQNotificationType Enumeration	-	The OracleAQNotificationType enumeration type specifies the notification type of the received notification.
OracleAQVisibilityMode Enumeration	-	The OracleAQVisibilityMode enumeration type specifies whether the enqueue or dequeue operation is part of the current transaction.
OracleBulkCopyOptions Enumeration	No	The OracleBulkCopyOptions enumeration specifies the values that can be combined with an instance of the OracleBulkCopy class and used as options to determine its behavior and the behavior of the WriteToServer methods for that instance.
OracleCollectionType Enumeration	-	OracleCollectionType enumerated values specify whether or not the OracleParameter object represents a collection, and if so, specifies the collection type.
OracleConnectionType Enumeration	No	OracleConnectionType enumerated values specify whether a particular connection object is associated with an Oracle database connection, a TimesTen database connection, or no physical connection at all.
OracleDBShutdownMode Enumeration	-	OracleDBShutdownMode enumerated values specify the database shutdown options.
OracleDBStartupMode Enumeration	-	OracleDBStartupMode enumerated values specify the database startup options.
OracleDbType Enumeration	-	OracleDbType enumerated values are used to explicitly specify the OracleDbType of an OracleParameter.
OracleDRCPYPurity Enumeration	-	OracleDRCPYPurity enumerated values specify the session purity.
OracleFailoverEvent Enumeration	-	OracleFailoverEvent enumerated values are used to specify the state of the failover.
<i>Not supported with unmanaged ODP.NET.</i>		

Table 1-2 (Cont.) Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Enumerations

Enumeration	Supported in the ODP.NET, Managed Driver and ODP.NET Core	Description
OracleFailoverRestore Enumeration	-	OracleFailoverRestore enumerated values indicate whether ODP.NET must restore session state after failover. <i>Not supported with unmanaged ODP.NET.</i>
OracleFailoverReturnCode Enumeration	-	OracleFailoverReturnCode enumerated values are passed back by the application to the ODP.NET provider to request a retry in case of a failover error, or to continue in case of a successful failover. <i>Not supported with unmanaged ODP.NET.</i>
OracleFailoverType Enumeration	-	OracleFailoverType enumerated values are used to indicate the type of failover event that was raised. <i>Not supported with unmanaged ODP.NET.</i>
OracleHAEventSource Enumeration	-	The OracleHAEventSource enumeration indicates the source of the HA event.
OracleHAEventStatus Enumeration	-	The OracleHAEventStatus enumeration indicates the status of the HA event source.
OracleIdentityType Enumeration	-	The OracleIdentityType enumeration specifies how Oracle identity column values are generated.
OracleNotificationInfo Enumeration	-	OracleNotificationInfo enumerated values specify the database event that causes the notification.
OracleNotificationSource Enumeration	-	OracleNotificationSource enumerated values specify the different sources that cause notification.
OracleNotificationType Enumeration	-	OracleNotificationType enumerated values specify the different types that cause the notification.
OracleParameterStatus Enumeration	-	The OracleParameterStatus enumeration type indicates whether a NULL value is fetched from a column, or truncation has occurred during the fetch, or a NULL value is to be inserted into a database column.
OracleRowidInfo Enumeration	-	The OracleRowidInfo enumeration values specify whether ROWID information is included as part of the ChangeNotificationEventArgs or not
OracleXmlCommandType Enumeration	-	The OracleXmlCommandType enumeration specifies the values that are allowed for the OracleXmlCommandType property of the OracleCommand class.

Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces

The `Oracle.DataAccess.Types` and `Oracle.ManagedDataAccess.Types` namespaces provides classes, structures, and exceptions for Oracle native types that can be used with Oracle Data Provider for .NET.

Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Structures

[Table 1-3](#) lists the type structures.

Table 1-3 Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Structures

Structure	Description
OracleBinary Structure	The <code>OracleBinary</code> structure represents a variable-length stream of binary data.
OracleBoolean Structure	The <code>OracleBoolean</code> structure represents a logical value that is either TRUE or FALSE.
OracleDate Structure	The <code>OracleDate</code> structure represents the Oracle DATE data type.
OracleDecimal Structure	The <code>OracleDecimal</code> structure represents an Oracle NUMBER in the database or any Oracle numeric value.
OracleIntervalDS Structure	The <code>OracleIntervalDS</code> structure represents the Oracle INTERVAL DAY TO SECOND data type.
OracleIntervalYM Structure	The <code>OracleIntervalYM</code> structure represents the Oracle INTERVAL YEAR TO MONTH data type.
OracleString Structure	The <code>OracleString</code> structure represents a variable-length stream of characters.
OracleTimeStamp Structure	The <code>OracleTimeStamp</code> structure represents the Oracle TIMESTAMP data type.
OracleTimeStampLTZ Structure	The <code>OracleTimeStampLTZ</code> structure represents the Oracle TIMESTAMP WITH LOCAL TIME ZONE data type.
OracleTimeStampTZ Structure	The <code>OracleTimeStampTZ</code> structure represents the Oracle TIMESTAMP WITH TIME ZONE data type.

Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Exceptions

Type Exceptions are thrown only by ODP.NET type structures. [Table 1-4](#) lists the type exceptions.

Table 1-4 Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Exceptions

Exception	Description
OracleTypeException Class	The <code>OracleTypeException</code> object is the base exception class for handling exceptions that occur in the ODP.NET Types classes.
OracleNullValueException Class	The <code>OracleNullValueException</code> represents an exception that is thrown when trying to access an ODP.NET Types structure that is null.

Table 1-4 (Cont.) Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Exceptions

Exception	Description
OracleTruncateException Class	The <code>OracleTruncateException</code> class represents an exception that is thrown when truncation in an ODP.NET Types class occurs.

Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Classes

[Table 1-5](#) lists the type classes.

Table 1-5 Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Classes

Class	Description
OracleArrayMappingAttribute Class	The <code>OracleArrayMappingAttribute</code> class is required to mark a custom class field or property with information that ODP.NET uses when a custom type represents an Oracle Collection type.
OracleBFile Class	An <code>OracleBFile</code> is an object that has a reference to BFILE data. It provides methods for performing operations on BFILE objects.
OracleBlob Class	An <code>OracleBlob</code> object is an object that has a reference to BLOB data. It provides methods for performing operations on BLOB objects.
OracleClob Class	An <code>OracleClob</code> is an object that has a reference to CLOB data. It provides methods for performing operations on CLOB objects.
OracleCustomTypeMappingAttribute Class	The <code>OracleCustomTypeMappingAttribute</code> class is used to mark a custom type factory class or struct with information that is used by ODP.NET when a custom type is used to represent an Oracle UDT.

Table 1-5 (Cont.) Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Classes

Class	Description
OracleObjectMappingAttribute Class	The OracleObjectMappingAttribute class marks custom class fields or properties with information that ODP.NET uses when a custom type represents an Oracle Object type.
OracleRef Class	An OracleRef instance represents an Oracle REF, which references a persistent, standalone, referenceable object that resides in the database. The OracleRef object provides methods to insert, update, and delete the Oracle REF.
OracleRefCursor Class	An OracleRefCursor object represents an Oracle REF CURSOR.
OracleUdt Class	The OracleUdt class defines static methods that are used when converting between Custom Types and Oracle UDTs and vice-versa.
OracleXmlStream Class	An OracleXmlStream object represents a sequential read-only stream of XML data stored in an OracleXmlType object.
OracleXmlType Class	An OracleXmlType object represents an Oracle XmlType instance.

Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Interfaces

[Table 1-6](#) lists the type interfaces.

Table 1-6 Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Interfaces

Interface	Description
IOracleArrayTypeFactory Interface	The IOracleArrayTypeFactory interface is used by ODP.NET to create arrays that represent Oracle Collections.

Table 1-6 (Cont.) Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Interfaces

Interface	Description
IOracleCustomType Interface	IOracleCustomType is an interface for converting between a Custom Type and an Oracle Object or Collection Type.
IOracleCustomTypeFactory Interface	The IOracleCustomTypeFactory interface is used by ODP.NET to create custom objects that represent Oracle Objects or Collections.
INullable Interface	The INullable interface is used to determine whether or not an ODP.NET type has a NULL value.

Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Enumerations

[Table 1-7](#) lists the type enumerations.

Table 1-7 Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Enumerations

Enumeration	Description
OracleUdtFetchOption Enumeration	OracleUdtFetchOption enumeration values specify how to retrieve a copy of the referenceable object.
OracleUdtStatus Enumeration	OracleUdtStatus enumeration values specify the status of an object attribute or collection element. An object attribute or a collection element can be a valid value or a null value.

Differences between the ODP.NET Drivers

ODP.NET, Managed Driver, ODP.NET, Unmanaged Driver, and ODP.NET Core have a number of configuration setting differences.

Table 1-8 Application Programming Interfaces not supported in ODP.NET, Managed Driver and ODP.NET Core

Namespace	Class/Enumeration/Interface	Unsupported Method/Property/Event
Oracle.ManagedDataAccess.Client	FailoverEvent enumeration	All
Oracle.ManagedDataAccess.Client	FailoverReturnCode enumeration	All
Oracle.ManagedDataAccess.Client	FailoverType enumeration	All
Oracle.ManagedDataAccess.Client	OracleAQAgent class	All
Oracle.ManagedDataAccess.Client	OracleAQDequeueMode enumeration	All
Oracle.ManagedDataAccess.Client	OracleAQDequeueOptions class	All
Oracle.ManagedDataAccess.Client	OracleAQEnqueueOptions class	All
Oracle.ManagedDataAccess.Client	OracleAQMessage class	All
Oracle.ManagedDataAccess.Client	OracleAQMessageAvailableEventArgs class	All
Oracle.ManagedDataAccess.Client	OracleAQMessageAvailableEventHandler class	All
Oracle.ManagedDataAccess.Client	OracleAQMessageDeliveryMode enumeration	All

Table 1-8 (Cont.) Application Programming Interfaces not supported in ODP.NET, Managed Driver and ODP.NET Core

Namespace	Class/Enumeration/Interface	Unsupported Method/Property/Event
Oracle.ManagedDataAccess.Client	OracleAQMessageState enumeration	All
Oracle.ManagedDataAccess.Client	OracleAQMessageType enumeration	All
Oracle.ManagedDataAccess.Client	OracleAQNavigationMode enumeration	All
Oracle.ManagedDataAccess.Client	OracleAQNotificationGroupingType enumeration	All
Oracle.ManagedDataAccess.Client	OracleAQNotificationType enumeration	All
Oracle.ManagedDataAccess.Client	OracleAQQueue class	All
Oracle.ManagedDataAccess.Client	OracleAQVisibilityMode enumeration	All
Oracle.ManagedDataAccess.Client	OracleClientFactory class	<i>Not supported in ODP.NET Core only</i>
Oracle.ManagedDataAccess.Client	OracleCommand class	ArrayBindRowsAffected property
Oracle.ManagedDataAccess.Client	OracleCommand class	ImplicitRefCursors property
Oracle.ManagedDataAccess.Client	OracleConfiguration class	<i>Most members not supported in ODP.NET, Unmanaged Driver only</i>
Oracle.ManagedDataAccess.Client	OracleConnection class	FlushCache() method
Oracle.ManagedDataAccess.Client	OracleConnection class	Failover event
Oracle.ManagedDataAccess.Client	OracleConnection class	ConnectionType property
Oracle.ManagedDataAccess.Client	OracleConnection class	EnlistDistributedTransaction(ITransaction) method <i>Not supported in ODP.NET Core only</i>
Oracle.ManagedDataAccess.Client	OracleConnection class	The properties BindByName, CommandTimeout, ChunkMigrationConnectionTimeout, FetchSize, KeepAlive, KeepAliveInterval, KeepAliveTime, MaxStatementCacheSize, Pipelining, ServiceRelocationConnectionTimeout, TnsAdmin, WalletLocation, and UseClientInitiatedCQN <i>Not supported in ODP.NET Unmanaged Driver only</i>
Oracle.ManagedDataAccess.Client	OracleConnectionType enumeration	All
Oracle.ManagedDataAccess.Client	OracleDataSourceCollection class	<i>Not supported in ODP.NET Unmanaged Driver only</i>
Oracle.ManagedDataAccess.Client	OracleDBShutdownMode enumeration	All
Oracle.ManagedDataAccess.Client	OracleDBStartupMode enumeration	All
Oracle.ManagedDataAccess.Client	OracleDataReader class	GetOracleBlobForUpdate() method If the method is called, then a NotSupportedException is thrown.

Table 1-8 (Cont.) Application Programming Interfaces not supported in ODP.NET, Managed Driver and ODP.NET Core

Namespace	Class/Enumeration/Interface	Unsupported Method/Property/Event
Oracle.ManagedDataAccess.Client	OracleDataReader class	GetOracleClobForUpdate() method If the method is called, then a NotSupportedException is thrown.
Oracle.ManagedDataAccess.Client	OracleDataReader class	IsAutoIncrement and IdentityType properties of the GetSchemaTable
Oracle.ManagedDataAccess.Client	OracleDataAdapter class	IdentityInsert property
Oracle.ManagedDataAccess.Client	OracleDataAdapter class	IdentityUpdate property
Oracle.ManagedDataAccess.Client	OracleDataAdapter class	SafeMapping property
Oracle.ManagedDataAccess.Client	OracleDatabase class	All
Oracle.ManagedDataAccess.Client	OracleException class	IsRecoverable property
Oracle.ManagedDataAccess.Client	OracleFailoverEventArgs class	All
Oracle.ManagedDataAccess.Client	OracleFailoverEventHandler class	All
Oracle.ManagedDataAccess.Client	OracleGlobalization class	ClientCharacterSet property
Oracle.ManagedDataAccess.Client	OracleGlobalization class	GetClientInfo() method
Oracle.ManagedDataAccess.Client	OracleGlobalization class	GetThreadInfo() method
Oracle.ManagedDataAccess.Client	OracleGlobalization class	SetThreadInfo() method
Oracle.ManagedDataAccess.Client	OracleIdentityType enumeration	All
Oracle.ManagedDataAccess.Client	OracleNotificationRequest class	GroupingInterval property
Oracle.ManagedDataAccess.Client	OracleNotificationRequest class	GroupingNotificationEnabled property
Oracle.ManagedDataAccess.Client	OracleNotificationRequest class	GroupingType property
Oracle.ManagedDataAccess.Client	OracleOnsServerCollection class	Not supported in ODP.NET Unmanaged Driver only
Oracle.ManagedDataAccess.Client	OraclePermission class	Not supported in ODP.NET Core only
Oracle.ManagedDataAccess.Client	OraclePermissionAttribute class	Not supported in ODP.NET Core only
Oracle.ManagedDataAccess.Client	OracleRowsCopiedEventArgs class	All
Oracle.ManagedDataAccess.Client	OracleRowsCopiedEventHandler class	All
Oracle.ManagedDataAccess.Client	IOracleCustomType interface	Equivalent functionality available in all providers with differences in parameters.

Table 1-8 (Cont.) Application Programming Interfaces not supported in ODP.NET, Managed Driver and ODP.NET Core

Namespace	Class/Enumeration/Interface	Unsupported Method/Property/Event
Oracle.ManagedDataAccess.Types	OracleRef class	The following unmanaged ODP.NET class members are <i>not</i> supported in managed and core: - Constructors - OracleRef(OracleConnecti on conn, string udtTypeName, string objTableName) - OracleRef(OracleConnecti on conn, string hexStr) - Methods - Flush() - GetCustomObject(OracleUdtFetchOption fetchOption, int depthLevel) - GetCustomObjectForUpdat e(bool bWait) - GetCustomObjectForUpdat e(bool bWait, int depthLevel) - Lock(bool bWait) - Properties - HasChanges {get} - ObjectTableName {get}
Oracle.ManagedDataAccess.Types	OracleTimestampTZ structure	OracleTimeStampTZ(DateTime dt, string timeZone) constructor. This constructor is supported but the timeZone must be an hour offset.
Oracle.ManagedDataAccess.Types	OracleUdt class	All providers have same APIs or equivalent APIs with differences in parameters.

See Also

[Oracle Data Provider for .NET, Managed Driver Configuration](#) and [Configuration Differences among ODP.NET Drivers](#) for more information about ODP.NET, Managed Driver .NET configuration settings.

[Distributed Transactions](#) for information about distributed transaction setup difference.

[Oracle Data Provider for .NET Core Configuration](#) for more information about configuring ODP.NET Core.

Getting Started With Developing ODP.NET Applications

Learn how to get started developing .NET applications for on-premises Oracle Database and Oracle Autonomous Database from one of the following links. These tutorials show how to create a simple .NET application that connects with either managed ODP.NET or ODP.NET Core. They also show how to use ODP.NET from Visual Studio and Visual Studio Code.

- [Developing .NET Applications for On-Premises Oracle Database](#)
- [Developing .NET Applications for Oracle Autonomous Database](#)
- Additional ODP.NET sample code is available in Oracle's .NET GitHub site:
<https://github.com/oracle/dotnet-db-samples>

2

Installing and Configuring Oracle Data Provider for .NET

This section describes installation and configuration requirements for Oracle Data Provider for .NET.

This section contains these topics:

- [System Requirements](#)
- [Entity Framework Requirements](#)
- [Entity Framework Core System Requirements](#)
- [Oracle Data Provider for .NET Versioning Scheme](#)
- [Installing Oracle Data Provider for .NET, Unmanaged Driver](#)
- [Installing Oracle Data Provider for .NET, Managed Driver](#)
- [Installing Oracle Data Provider for .NET Core](#)
- [Entity Framework Assemblies and File Location](#)
- [Configuring Oracle Data Provider for .NET](#)
- [Oracle Data Provider for .NET, Unmanaged Driver Configuration](#)
- [Oracle Data Provider for .NET, Managed Driver Configuration](#)
- [Oracle Data Provider for .NET Core Configuration](#)
- [Configuration Differences among ODP.NET Drivers](#)
- [Centralized Configuration Providers for Deployments](#)
- [Connection Configuration Restriction](#)
- [Configuring for Entity Framework Code First](#)
- [Migrating from ODP.NET, Unmanaged Driver to ODP.NET, Managed Driver or ODP.NET Core](#)
- [Configuring a Port to Listen for Database Notifications](#)
- [General .NET Programming Recommendations and Tips for ODP.NET](#)

System Requirements

Oracle Data Provider for .NET, Unmanaged Driver requires the following:

- Windows operating system
 - 64-bit: Windows 10 x64 (Pro, Enterprise, and Education Editions), Windows 11 x64 (Pro, Enterprise, and Education Editions), Windows Server 2016 x64 (Standard, Datacenter, and Essentials Editions), Windows Server 2019 (Standard, Datacenter, and Essentials Editions), Windows Server 2022 x64 (Standard, Datacenter, and Essentials Editions), or Windows Server 2025 x64 (Standard, Datacenter, and Essentials Editions).

Oracle supports 64-bit ODP.NET for Windows x64 on these operating systems.

- Microsoft .NET
 - ODP.NET for .NET Framework 4 is only supported with Microsoft .NET Framework 4.7.2 and higher.
- Access to Oracle Database 19c or later
- Oracle Client release 26ai

This is automatically installed as part of the ODP.NET installation.

Promotable and distributed transactions may require Oracle Services for Microsoft Transaction Server in whole or in part. Refer to the [Distributed Transactions](#) section for more information.

Oracle Data Provider for .NET, Managed Driver requires the following:

- Same Windows operating system support as ODP.NET, Unmanaged Driver.
ODP.NET, Managed Driver only runs on 64-bit .NET Framework and 64-bit (x64) Windows.
- Microsoft .NET Framework 4.7.2 and higher.
- Access to Oracle Database 19c or later

ODP.NET Core requires the following:

- Operating systems:
 - Same Windows operating system support as managed and unmanaged ODP.NET
 - Oracle Linux 9.2 or higher
 - Oracle Linux 8.6 or higher
 - Red Hat Enterprise Linux 9.2 or higher
 - Red Hat Enterprise Linux 8.6 or higher
 - 64-bit Ubuntu 22.04.1 LTS 5.15.0-1016 or higher
 - 64-bit Debian GNU/Linux 11 (bullseye) 5.10.120-1 or higher
 - Alpine Linux: 3.20.3 or higher
 - macOS (ARM64) 14.7 or higher
- Microsoft .NET
 - ODP.NET Core supports .NET 8.
 - ODP.NET Core 23.6.1 and higher supports .NET 9.
 - ODP.NET Core 23.26.0 and higher support .NET 10.
- Access to Oracle Database 19c or later

ODP.NET Core is compatible with ASP.NET Core.

ODP.NET Core supports only 64-bit .NET.

① See Also

- Document 726240.1 on My Oracle Support (formerly OracleMetaLink) for details on supported configuration for different ODP.NET versions. You can access My Oracle Support from:

<https://support.oracle.com/portal/>

- <https://learn.microsoft.com/en-us/docs/>

Entity Framework Requirements

This section contains the following topics:

- [Entity Framework Database First and Model First Requirements](#)
- [Entity Framework Code First Requirements](#)

Entity Framework Database First and Model First Requirements

Oracle's support for Entity Framework Database First and Model First has the following version requirement:

- Microsoft Entity Framework 6.5.1
- Microsoft .NET Framework 4.8.x
- ODP.NET, Managed Driver

If using Visual Studio tools, then install Oracle Developer Tools for Visual Studio.

Entity Framework Code First Requirements

Oracle's support for Entity Framework Code First has the following version requirements:

- Microsoft Entity Framework 6.5.1
- Microsoft .NET Framework 4.8.x

Projects must set the target framework to a supported .NET Framework version. This can be done by modifying the project's properties in Visual Studio.

- ODP.NET, Managed Driver

Entity Framework Core System Requirements

ODP.NET Entity Framework Core (EF Core) has the same system requirements as ODP.NET Core. In addition, ODP.NET EF Core has the following dependencies:

- ODP.NET Core
- Entity Framework Core 8, 9 and 10. Oracle Entity Framework Core 23.6 and higher supports Entity Framework 9. Oracle Entity Framework Core 23.26.0 and higher support Entity Framework Core 10.
- `Microsoft.EntityFrameworkCore.Relational`

Oracle Data Provider for .NET Versioning Scheme

Oracle Data Provider for .NET, Unmanaged Driver; ODP.NET, Managed Driver; and ODP.NET Core each ship with their own set of binaries.

For example, 26ai binaries would be the following:

- Unmanaged ODP.NET for .NET Framework 4
 - `Oracle.DataAccess.dll`
 - * Built with .NET Framework 4
 - * Assembly version number: 4.122.23.1
 - `OraOps.dll` and Oracle Client
 - * Used by ODP.NET for .NET Framework
- Managed ODP.NET for .NET Framework 4:
 - `Oracle.ManagedDataAccess.dll`
 - * Built with .NET Framework 4
 - * Assembly version number: 4.122.23.1
- ODP.NET Core:
 - `Oracle.ManagedDataAccess.dll`
 - * Built with .NET (Core) version
 - * Assembly version number: 23.1.0.0

The convention for managed and unmanaged ODP.NET assembly/DLL product versioning is *n1.o1o2.o3.o4:o5*

where

- *n1* is the most significant .NET Framework version number.
- *o1o2* once represented the first two digits of the ODP.NET 12.2 version number. It remains "122" as modifying this version would not allow in place ODP.NET upgrades without an application rebuild.
- *o3* is the first digit of the ODP.NET product version number.
- *o4* indicates whether the release is a production version (1) or beta/pre-release version (0).
- *o5* is the ODP.NET assembly build date.

For example, if the ODP.NET version number is 23.4, the corresponding managed and unmanaged ODP.NET assembly product version is 4.122.23.1:20240502.

For ODP.NET Core, the assembly product version convention is:

n1.n2.o1.o2:o3

- *n1* is the first digit of the ODP.NET product version number.
- *n2* indicates whether the release is a production version (1) or beta/pre-release version (0).
- *o1* is currently unused.
- *o2* is currently unused.

- o3 is the ODP.NET assembly build date.

For example, if the ODP.NET Core version number is 23.4, the corresponding assembly product version is 23.1.0.0.20240502.

The ODP.NET *assembly* version is distinct from the *assembly product* version. The *assembly* version uses four sets of digits and the *assembly product* version uses five sets of digits. For each ODP.NET release, the first four sets of digits of the assembly version and the assembly product version will be the same. For example, if the ODP.NET Core assembly product version is 23.1.0.0.20240502, then its assembly version is 23.1.0.0. In the case of managed and unmanaged ODP.NET 4.122.23.1:20240502, the assembly version is 4.122.23.1.

The assembly product version identifies the precise ODP.NET version used. Since there are multiple updates Oracle ships within a major release family, you would provide the ODP.NET assembly product version in a support context.

The assembly version is used in more .NET-specific contexts. For example, the <version> section of the .NET configuration file uses the four digit assembly version to identify which ODP.NET version to use.

The assembly product version number can be found in the ODP.NET DLL "Product version" property on Windows. The assembly version can be found in the ODP.NET DLL "File version" property on Windows.

Note that the Oracle installer and documentation still refer to the ODP.NET product version number and not the assembly/DLL version number.

Publisher Policy DLL is provided as before so that applications built with older versions of ODP.NET are redirected to the newer ODP.NET assembly, even though the versioning scheme has changed.

ODP.NET, Managed Driver Versioning

ODP.NET, Managed Driver uses assembly manifest attribute `AssemblyInformationalVersionAttribute` to uniquely identify assemblies with the same `AssemblyVersionAttribute` attribute value. This value can be accessed via .NET code, PowerShell, and other Windows applications to identify ODP.NET, Managed Driver versions uniquely.

`AssemblyInformationalVersionAttribute` is set to the same version as the actual .NET assembly version.

This value is accessible using .NET Framework `System.Diagnostics.FileVersionInfo.ProductVersion` property. The returned value can be used as a `Version` object or as a comparison String using comparison operators or methods. Essentially, among a collection of ODP.NET, Managed Driver assemblies that have the same assembly version, the newest ODP.NET, Managed Driver assembly will have the largest fourth digit `ProductVersion` value than an older assembly.

PowerShell Example: In this example, administrators uniquely distinguish the assemblies between ODP.NET, Managed Driver versions from an old version of ODP.NET, Managed Driver in `c:\old` and a more recent one in `c:\new`.

Script:

```
$VC1 = New-Object System.Version((Get-Command
C:\old\Oracle.ManagedDataAccess.dll).FileVersionInfo.ProductVersion)
$VC2 = New-Object System.Version((Get-Command
C:\new\Oracle.ManagedDataAccess.dll).FileVersionInfo.ProductVersion)
"Compare V1 to V2: " + $VC1.CompareTo($VC2)
```

```
"Compare V1 to V1: " + $VC1.CompareTo($VC1)
"Compare V2 to V1: " + $VC2.CompareTo($VC1)
```

Output:

```
Compare V1 to V2: -1
Compare V1 to V1: 0
Compare V2 to V1: 1
```

Note

ProductVersion property comparisons will provide correct information on which version is more recent than the other *only* for ODP.NET, Managed Driver.

Installing Oracle Data Provider for .NET, Unmanaged Driver

Oracle Data Provider for .NET, Unmanaged Driver can be installed through Oracle AI Database software, a Microsoft installer (MSI), or XCopy. One example of an MSI is Oracle Client for Microsoft Tools. Xcopy is a command-line Oracle Data Access Components (ODAC) software installation. Unmanaged ODP.NET software is available from Oracle's website for download. This section provides details on unmanaged ODP.NET installation.

Administrators use XCopy to deploy Oracle Data Provider for .NET to large numbers of computers for production deployments. The XCopy has a smaller installation size and fine-grain control during installation and configuration than Oracle Universal Installer.

Note

This section describes non-ODAC installation using the Oracle Universal Installer for database server or client. For installation and configuration using the XCopy install, refer to the README.TXT file that is part of the XCopy installation.

Additionally, Oracle Data Provider for .NET Dynamic Help is registered with Visual Studio, providing context-sensitive online help that is seamlessly integrated with Visual Studio Dynamic Help. With Dynamic Help, the user can access ODP.NET documentation within the Visual Studio IDE by placing the cursor on an ODP.NET keyword and pressing the F1 function key.

See Also

[Configuring for Entity Framework Code First](#)

[Oracle Client for Microsoft Tools](#)

File Locations After Installation

The Oracle.DataAccess.dll assembly is installed to the following location:

.NET Framework 4:

`ORACLE_HOME\odp.net\bin\4` directory

Search Order for Unmanaged DLLs

Unmanaged ODP.NET consists of managed and unmanaged binaries. Through the use of the `DllPath` configuration parameter, each application can specify the `ORACLE_HOME\bin` location that the dependent unmanaged Oracle Client binaries are loaded from. However, the `ORACLE_HOME` must have the same unmanaged ODP.NET version installed as the version that the application uses. Otherwise, a version mismatch exception is thrown.

The `Oracle.DataAccess.dll` searches for dependent unmanaged DLLs (such as Oracle Client) based on the following order:

1. Directory of the application or executable.
2. `DllPath` setting specified by application config or `web.config`.
3. `DllPath` setting specified by `machine.config`.
4. `DllPath` setting specified by the Windows Registry.
`HKEY_LOCAL_MACHINE\Software\Oracle\ODP.NET\version\DllPath`
5. Directories specified by the Windows PATH environment variable.

Upon installation of unmanaged ODP.NET, Oracle Universal Installer sets the `DllPath` Windows Registry value to the `ORACLE_HOME\bin` directory where the corresponding dependent DLLs are installed. Developers must provide this configuration information on an application-by-application basis.

Note

`Oracle.DataAccess.dll` uses the unmanaged DLL, `OraOpsXX.dll`, where the `XX` is replaced by the version number.

Unmanaged ODP.NET and Dependent Unmanaged DLL Mismatch

To enforce the usage of `Oracle.DataAccess.dll` assembly with the correct version of its unmanaged DLLs, an exception is raised if `Oracle.DataAccess.dll` notices it has loaded a mismatched version of a dependent unmanaged DLL.

Installing Oracle Data Provider for .NET, Managed Driver

Getting started with ODP.NET, Managed Driver

Get started with ODP.NET, Managed Driver by either using NuGet or a Windows Installer.

If you are using NuGet: Download the ODP.NET NuGet package and use NuGet Package Manager to install.

If you are using Windows Installer: Follow the VSIX or Microsoft Windows Installer (MSI) steps to install ODP.NET, Managed Driver.

ODP.NET, Managed Driver Files

ODP.NET, Managed Driver consists of the following files:

Table 2-1 ODP.NET, Managed Driver Files with Descriptions

File	Description
Oracle.ManagedDataAccess.dll	Platform-independent (AnyCPU), fully-managed ADO.NET provider
\Resources\<lang>\Oracle.ManagedDataAccess.resources.dll	Platform-independent (AnyCPU), fully-managed ADO.NET provider resource DLLs.
OraProvCfg.exe	Platform-independent (AnyCPU) utility to configure/unconfigure ODP.NET, Managed and Unmanaged Drivers.
configure.bat	Batch file to add managed ODP.NET Windows Registry entries.
unconfigure.bat	Batch file to remove managed ODP.NET Windows Registry entries.
tnsnames.ora	A sample configuration file that defines data source aliases.
sqlnet.ora	A sample configuration file that configures network related settings.
Oracle.ManagedDataAccess.Client.Configuration.Section.xsd	An XML schema file that defines the configuration section for ODP.NET, Managed Driver.
Oracle.ManagedDataAccess.EntityFramework.dll	Platform-independent (AnyCPU), fully-managed assembly for Code First and Entity Framework 6 higher
\x64\Oracle.ManagedDataAccessIOP.dll	Platform-dependent (64-bit .NET Framework), Managed Assembly for MIT Kerberos support

The OraProvCfg.exe utility can perform many different configuration tasks, such as GACing ODP.NET or configuring ODP.NET in the machine.config file. To learn about how to use this utility, run oraprovcfg.exe on the command line without any parameters, which will output the documentation.

See Also

Oracle AI Database Installation Guide for Microsoft Windows for installation instructions

Platform-Dependent Assemblies and Their Search Order

ODP.NET, Managed Driver has one platform-dependent DLL: Oracle.ManagedDataAccessIOP.dll, which is 64-bit. While it consists of 100% managed code, it calls APIs outside of .NET, which is why it is platform dependent.

Oracle.ManagedDataAccessIOP.dll supports MIT Kerberos. The assembly is only needed in your application if you are using MIT Kerberos.

The assembly is not intended to be directly referenced by an application. Rather, it is referenced implicitly. ODP.NET, Managed Driver references the assembly by using the following search order:

1. Global Assembly Cache

2. The web application's bin directory or Windows application's EXE directory
3. The DLL's default install location. Oracle recommends using this method of finding dependent assemblies if your application is AnyCPU.

For example, use the following steps for your application to use `Oracle.ManagedDataAccessIOP.dll`:

1. Right click **Visual Studio project**, select **Add**, and then select **New Folder**.
2. Name the folder x64.
3. Right-click the newly created **x64** folder, select **Add**, and then select **Existing Item**.
4. Browse to the folder where the DLL is located, which usually is `ORACLE_HOME\odp.net\managed\x64`, and then select **Oracle.ManagedDataAccessIOP.dll**.
5. Click **Add**.
6. Click the newly added **Oracle.ManagedDataAccessIOP.dll** in the x64 folder.
7. In the properties window, set **Copy To Output Directory** to **Copy Always**.

For x86 targeted applications, name the folder x86 and add the assembly from the x86 directory.

To make your application platform independent even if it depends on `Oracle.ManagedDataAccessIOP.dll` create both x64 and x86 folders with the necessary assemblies added to them.

File Locations After Installation

In an Oracle Universal Installer based install, the `Oracle.ManagedDataAccess.dll` assembly is installed to the following location:

.NET Framework 4:

`ORACLE_HOME\ODP.NET\nuget` directory

Managed ODP.NET is embedded within the `Oracle.ManagedDataAccess.<version>.nupkg` file. This file is a NuGet package. ODP.NET users can install this package using common Microsoft tools, such as NuGet Package Manager and Visual Studio.

When Oracle Developer Tools for Visual Studio is installed, Oracle documentation is installed in Visual Studio itself.

Samples are provided in GitHub:

<https://github.com/oracle/dotnet-db-samples>

Installing Oracle Data Provider for .NET Core

Getting started with ODP.NET Core

Get started with ODP.NET Core on Windows by using NuGet, Windows Installer, or Oracle Universal Installer (OUI). Only NuGet can be used to install ODP.NET Core on non-Windows operating systems.

If you are using NuGet: Download the ODP.NET Core NuGet package and use NuGet Package Manager to install.

If you are using Windows Installer: Follow the Microsoft Windows Installer (MSI) steps. This installer will copy the ODP.NET Core NuGet package to the machine.

If you are using OUI: Follow the OUI steps to install ODP.NET Core. OUI will copy the ODP.NET Core NuGet package to the machine.

ODP.NET Core Files

ODP.NET Core consists of the following files:

Table 2-2 ODP.NET, Core Files with Descriptions

File	Description
<code>Oracle.ManagedDataAccess.dll</code>	Platform-independent (AnyCPU), operating system independent, fully-managed ADO.NET provider

File Locations After Installation

In an Oracle Universal Installer based install, the ODP.NET Core package is installed to the following directory:

`ORACLE_HOME\ODP.NET\nuget`

ODP.NET Core is embedded within the `Oracle.ManagedDataAccess.Core.<version>.nupkg` file. This file is a NuGet package. ODP.NET users can install this package using common Microsoft tools, such as NuGet Package Manager and Visual Studio."

Samples are provided on our ODP.NET GitHub site:

<https://github.com/oracle/dotnet-db-samples>

Entity Framework Assemblies and File Location

ODP.NET ships with a separate assembly to support Code First and Entity Framework 6. This model physically separates Entity Framework 6 functionality from ADO.NET functionality.

The managed ODP.NET assembly is `Oracle.ManagedDataAccess.EntityFramework.dll`.

Unmanaged ODP.NET discontinued Entity Framework support starting with ODP.NET 21c. Unmanaged ODP.NET EF apps should migrate to managed ODP.NET.

When installed using the XCopy package, the Oracle Entity Framework assembly is found in the following location after install:

`%ORACLE_HOME%\odp.net4\odp.net\managed\common\EF6`

where `%ORACLE_HOME%` represents the operating system path to the installation directory.

The assembly is fully managed code and is designed to be bin deployable meaning that the assembly should be copied into the application's `bin` directory. As such the assembly is not registered in the Global Assembly Cache (GAC) during installation.

In Oracle Universal Installer, the Oracle Entity Framework (Core) assemblies are now NuGet packages located in:

`%ORACLE_HOME%\ODP.NET\nuget`

In that location, you will find managed ODP.NET Entity Framework 6 and ODP.NET Entity Framework Core NuGet packages. ODP.NET users can install these packages using common Microsoft tools, such as NuGet Package Manager and Visual Studio.

Note

If desired the Oracle Entity Framework 6 assemblies may be registered in the GAC manually but Oracle recommends not doing so.

Configuring Oracle Data Provider for .NET

The settings for specific versions of ODP.NET, can be configured in several ways for specific effects on precedence:

- The Windows registry entries are machine-wide settings for a particular version of ODP.NET.
Windows registry based configuration is not supported for ODP.NET, Managed Driver nor ODP.NET Core.
- The machine.config settings are .NET framework-wide settings that override the Windows registry values.
- The application or web config file settings are application-specific settings that override the machine.config settings and the Windows registry settings.

Note

There is one exception to app/web/config settings overriding machine.config. For `oracle.manageddataaccess.client` and `oracle.unmanageddataaccess.client` sections, a machine.config with a specific ODP.NET version subsection, that is, `<version number="4.121.2.0">`, will override an app/web.config subsection that references all versions generically, that is, `<version number="*">`. To override the machine.config subsection, create a subsection for that version in the app/web/config file, that is, `<version number="4.121.2.0">`.

- Any attribute settings made in the ODP.NET application code override everything else.

The application or web config file can be useful and sometimes essential in scenarios where more than one application on a computer use the same version of ODP.NET, but each application needs a different ODP.NET configuration. The Windows registry value settings for a given version of ODP.NET affect all the applications that use that version of ODP.NET. However, having ODP.NET configuration values in the application or web config file assure that these settings are applied only for that application, thus providing more granularities.

For example, if the application or web.config file has a `StatementCacheSize` configuration setting of 100, this application-specific setting forces the version of ODP.NET that is loaded by that application to use 100 for the `StatementCacheSize` and overrides any setting in the machine.config and in the registry. Note that for any setting that does not exist in a config file (machine.config or application/web config), the value in the registry for a loaded version of ODP.NET is used, as in previous releases.

Note that ODP.NET reads the `machine.config` files from the version of the .NET Framework on which ODP.NET runs, not from the version of ODP.NET.

ODP.NET only reads the Windows Registry and the XML configuration file when it is loaded into memory, thus any configuration changes made after that are not read or used until the application is re-started.

All boolean attributes in ODP.NET .NET configuration settings accept `true`, `false`, `1`, and `0` as valid values. `1` is equivalent to `true` and `0` is equivalent to `false`.

Note

ODP.NET Core does not support the Windows registry nor .NET configuration files. ODP.NET Core configuration can be set using .NET Configuration API, `sqlnet.ora` file, and `tnsnames.ora` file.

Oracle Client Configuration File Automated Setup During Installation

When installing Oracle Data Access Components (ODAC) in a new Oracle Home, Oracle Universal Installer (OUI) automatically copies the Oracle local naming (`tnsnames.ora`), profile (`sqlnet.ora`), and directory (`ldap.ora`) parameter files and settings from an existing Oracle home into the newly installed ODAC home, as long as they share the same bitness. That is, they are both 32-bit installations or they are both 64-bit installations.

Alternatively, existing `*.ora` files can be copied over from another existing Oracle home, besides the last active one, to the new ODAC Oracle home. OUI provides location information for these files from up to three other existing Oracle homes if they exist. The `*.ora` files can be customized if the new Oracle home uses a different configuration from the previous Oracle home from which the files were copied over.

If you install into an existing ODAC or RDBMS Oracle home, then no new `*.ora` files is copied or created.

If you install onto a computer without any previous Oracle homes present, then OUI prompts the user for the database connection alias information. OUI then automatically creates the `tnsnames.ora` file. If no alias information is provided, then no `tnsnames.ora` file is created. Even if the user does not have all the database connection information readily available, Oracle recommends inserting placeholder values during the install process, then modifying the `tnsnames.ora` file later with actual values to replace the placeholders.

Oracle Client Configuration File Settings

ODP.NET configuration file parameter values can be set in .NET configuration, `tnsnames.ora`, `sqlnet.ora`, and `ldap.ora` files. The `*.ora` file location can be a location different from the standard `ORACLE_HOME/network/admin` directory. The `*.ora` settings order of precedence is similar to ODP.NET's settings order of precedence. The main difference is that the `*.ora` files themselves are included in the search order. The ODP.NET connection configuration precedence order is as follows.

Managed ODP.NET:

1. `OracleConfiguration.OracleDataSources`
2. `<dataSources>` in .NET configuration file
3. Directory set in `OracleConnection.TnsAdmin` property

4. Directory set for the Tns_Admin (connection string attribute) or tns_admin setting (connection string URL or centralized configuration file connection string).
5. Directory set in OracleConfiguration.TnsAdmin property
6. TNS_ADMIN directory setting in .NET configuration file
7. Current working directory
8. TNS_ADMIN directory setting of the Windows environment variable or container environment variable
9. Directory set in the Windows Registry key, HKLM\SOFTWARE\Oracle\ODP.NET.Managed\<version>\TNS_ADMIN for 64-bit apps

Note

32-bit apps use the Registry key
HKLM\SOFTWARE\Wow6432Node\Oracle\ODP.NET.Managed\<version>\TNS_ADMIN

10. %ORACLE_HOME%\network\admin directory

Unmanaged ODP.NET:

1. <dataSources> and <settings> in .NET configuration file
2. Current working directory
3. TNS_ADMIN directory setting of the Windows environment variable or container environment variable
4. TNS_ADMIN Windows Registry setting (HKLM\SOFTWARE\ORACLE\KEY_<Oracle Version>HOME<#>\)
5. %ORACLE_HOME%\network\admin directory

The managed ODP.NET ldap.ora precedence order is as follows:

1. <LDAPsettings> and <settings> in .NET configuration file
2. Directory set in OracleConnection.TnsAdmin property
3. Directory set for the Tns_Admin connection string attribute
4. Directory set in OracleConfiguration.TnsAdmin property
5. Directory set in OracleConfiguration.LdapAdmin property
6. TNS_ADMIN directory setting in .NET configuration file
7. LDAP_ADMIN directory setting in .NET configuration file
8. Current working directory
9. TNS_ADMIN directory setting in the Windows environment variable
10. LDAP_ADMIN directory setting in the Windows environment variable

The unmanaged ODP.NET ldap.ora precedence order is as follows:

1. LDAP_ADMIN directory setting in the Windows environment variable
2. %ORACLE_HOME%\ldap\admin directory
3. TNS_ADMIN directory setting in the Windows environment variable
4. %ORACLE_HOME%\network\admin directory

Oracle recommends using an `app.config` or `web.config` file to store all these Oracle Client configuration parameter settings.

Once the first `tnsnames.ora`, `sqlnet.ora`, and `ldap.ora` are found and read, no additional `*.ora` file lower in the precedence order is read. That means all Oracle Client configuration settings must be made in the `app.config`, `web.config`, `machine.config`, or the first set of `*.ora` files found. Additional parameter values set in `*.ora` files lower in the precedence order will not be read.

Configuring .NET Framework to Use ODP.NET

Automatic ODP.NET machine-wide configuration upon installation is no longer available as of ODAC 18c. This change improves compatibility when applications on the same machine use different ODP.NET versions and/or depend on application-specific configuration settings instead of machine-wide assembly registration and configuration.

Nonetheless, administrators can still manually place managed and unmanaged ODP.NET in the Global Assembly Cache (GAC), as well as add configurations for section handler, `DbProviderFactories`, `TNS_ADMIN`, `LDAP_ADMIN`, and assembly redirection (in place of policy DLLs), into `machine.config`, if desired.

Most ODP.NET settings in `machine.config` can be overridden by local settings in `app.config` or `web.config`. However, conflicts between the `machine.config` and, `app.config` or `web.config` can prevent applications from running properly. Care should be taken when configuring ODP.NET at both the `machine.config` level and application level using `app.config` or `web.config`.

To avoid such issues, Oracle recommends ODP.NET application configuration settings reside in the `app.config` or `web.config`. The following sections explore how to set up these ODP.NET configuration sections and provide sample configuration sections and files.

Configuration Section Handler

In order for the application to read the ODP.NET configuration section, a section handler must be configured. The following entry should be added to the .NET configuration file to enable applications to add an `oracle.manageddataaccess.client` section for ODP.NET, Managed Driver-specific configuration:

```
<configuration>
  <configSections>
    <section name="oracle.manageddataaccess.client"
      type="OracleInternal.Common.ODPMSectionHandler, Oracle.ManagedDataAccess,
      Version=4.122.19.1, Culture=neutral, PublicKeyToken=89b483f429c47342" />
    </configSections>
  </configuration>
```

Note

In all the examples in this section, the version of the ODP.NET assembly is understood to be 4.122.19.1. This version number should be modified appropriately if you are using another version.

The unmanaged ODP.NET equivalent section handler is as follows:

```
<configuration>
  <configSections>
```

```

    <section name="oracle.unmanageddataaccess.client"
type="OracleInternal.Common.CustomSectionHandler, Oracle.DataAccess, Version=4.122.19.1,
Culture=neutral, PublicKeyToken=89b483f429c47342" />
  </configSections>
</configuration>

```

One of the configuration entries that either has to match between machine.config and, app.config or web.config, or only exist in the app.config or web.config to avoid any conflict is the configuration for the section handler. For example, if your application is a web application and the above entry was added to a web.config and the same configuration section handler for oracle.manageddataaccess.client also exists in machine.config but the Version attribute values are different, an error message of "There is a duplicate oracle.manageddataaccess.client section defined." may be observed at runtime. To avoid this issue, the configuration section handler entry in the machine.config for oracle.manageddataaccess.client has to be removed from the machine.config or the ODP.NET config section handler entry in machine.config and web.config has to match exactly. If there are other applications on the machine that depend on this entry in the machine.config, then this section handler entry will need to be moved to all local applications' .NET configuration files.

DbProviderFactories

The following entry should be added in the app.config or web.config for applications that use DbProviderFactories and DbProviderFactory classes. Also, any DbProviderFactories entry for Oracle.ManagedDataAccess.Client in the machine.config will be ignored due to the <remove> entry:

```

<configuration>
  <system.data>
    <DbProviderFactories>
      <remove invariant="Oracle.ManagedDataAccess.Client" />
      <add name="ODP.NET, Managed Driver" invariant="Oracle.ManagedDataAccess.Client"
description="Oracle Data Provider for .NET, Managed Driver"
type="Oracle.ManagedDataAccess.Client.OracleClientFactory, Oracle.ManagedDataAccess,
Version=4.122.19.1, Culture=neutral, PublicKeyToken=89b483f429c47342" />
    </DbProviderFactories>
  </system.data>
</configuration>

```

The equivalent for unmanaged ODP.NET is as follows:

```

<configuration>
  <system.data>
    <DbProviderFactories>
      <remove invariant="Oracle.DataAccess.Client" />
      <add name="Oracle Data Provider for .NET" invariant="Oracle.DataAccess.Client"
description="Oracle Data Provider for .NET"
type="Oracle.DataAccess.Client.OracleClientFactory, Oracle.DataAccess,
Version=4.122.19.1, Culture=neutral, PublicKeyToken=89b483f429c47342" />
    </DbProviderFactories>
  </system.data>
</configuration>

```

Dependent Assembly

For users accustomed to the policy DLLs that were registered automatically upon installation, the same can be accomplished through configuration. The following is an example configuration that redirects the application that depends on 4.122 version of ODP.NET and higher to be redirected to use ODP.NET 4.122.19.1. Note that in this example, an entry is added to ignore policy ODP.NET DLLs that are registered in the GAC:

```
<configuration>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <publisherPolicy apply="no" />
        <assemblyIdentity name="Oracle.ManagedDataAccess"
publicKeyToken="89b483f429c47342" culture="neutral" />
        <bindingRedirect oldVersion="4.122.0.0 - 4.65535.65535.65535"
newVersion="4.122.19.1" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

The equivalent for unmanaged ODP.NET is as follows:

```
<configuration>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <publisherPolicy apply="no" />
        <assemblyIdentity name="Oracle.DataAccess" publicKeyToken="89b483f429c47342"
culture="neutral" />
        <bindingRedirect oldVersion="4.122.0.0 - 4.65535.65535.65535"
newVersion="4.122.19.1" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

The final .NET configuration section that includes a configuration section handler, DbProviderFactories, dependent assembly, and provider configuration would look similar to the following example:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <section name="oracle.manageddataaccess.client"
      type="OracleInternal.Common.ODPMSectionHandler, Oracle.ManagedDataAccess,
Version=4.122.19.1, Culture=neutral, PublicKeyToken=89b483f429c47342"/>
    </configSections>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.1"/>
  </startup>
  <system.data>
    <DbProviderFactories>
      <remove invariant="Oracle.ManagedDataAccess.Client"/>
      <add name="ODP.NET, Managed Driver"
invariant="Oracle.ManagedDataAccess.Client" description="Oracle Data Provider for .NET,
Managed Driver"
type="Oracle.ManagedDataAccess.Client.OracleClientFactory, Oracle.ManagedDataAccess,
Version=4.122.19.1, Culture=neutral, PublicKeyToken=89b483f429c47342"/>
    </DbProviderFactories>
  </system.data>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <publisherPolicy apply="no"/>
        <assemblyIdentity name="Oracle.ManagedDataAccess"
publicKeyToken="89b483f429c47342" culture="neutral"/>
        <bindingRedirect oldVersion="4.121.0.0 - 4.65535.65535.65535"
newVersion="4.122.19.1" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

```

newVersion="4.122.19.1"/>
    </dependentAssembly>
    </assemblyBinding>
</runtime>
<oracle.manageddataaccess.client>
    <version number="*">
        <dataSources>
            <dataSource alias="SampleDataSource"
descriptor="(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=localhost)(PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=ORCL)))" />
        </dataSources>
    </version>
</oracle.manageddataaccess.client>
</configuration>

```

ODP.NET Intellisense for .NET Configuration Files

When installing ODP.NET from a non-ODAC release, such as from a database server installation, Intellisense for ODP.NET configuration files is not enabled by default as it is with ODAC-based GUI installations. .xsd files are included in the ORACLE_HOME\odp.net\managed\common directory to enable developers to modify the .NET configuration file using IntelliSense. For Visual Studio to leverage the supplied .xsd files, do the following:

1. Open the .NET config file (i.e. app.config, web.config, or machine.config) within Visual Studio.
2. Within Visual Studio, navigate to **XML**, and then **Schemas**.
3. Enable the check mark under the **Use** column for:
 - Oracle.ManagedDataAccess.Client.Configuration.Section.xsd for managed ODP.NET and
 - Oracle.UnmanagedDataAccess.Client.Configuration.Section.xsd for unmanaged ODP.NET, respectively.

Oracle Data Provider for .NET, Unmanaged Driver Configuration

The following sections explain how to configure ODP.NET, Unmanaged Driver.

ODP.NET can be configured using an XML file named web.config, app.config, or machine.config. These config files contain sections specific to ODP.NET configuration.

For unmanaged ODP.NET, developers use either the traditional <oracle.dataaccess.client> section or the newer <oracle.unmanageddataaccess.client> section. Oracle recommends applications use <oracle.unmanageddataaccess.client> when possible. For managed ODP.NET, developers use <oracle.manageddataaccess.client>.

<oracle.unmanageddataaccess.client> is a superset of <oracle.manageddataaccess.client> as unmanaged ODP.NET supports some features not available in the managed driver. For features both providers have in common, they share the same structure, properties, and nearly all values. Programmers will find using either provider interchangeably or migrating between unmanaged and managed ODP.NET is easier with the shared format.

This documentation section covers unmanaged ODP.NET configuration settings in the Windows registry, <oracle.dataaccess.client>, or unique

<oracle.unmanageddataaccess.client> settings. For shared settings with <oracle.manageddataaccess.client>.

See Also

[Oracle Data Provider for .NET, Managed Driver Configuration](#)

Supported Configuration Settings

ODP.NET, Unmanaged Driver supports the configuration of an attribute as follows:

- In the Windows registry.
- In an XML file.
- Through a different mechanism such as a connection string or programmatically through an ODP.NET class, if applicable.

[Table 2-3](#) describes each configurable attribute that is supported by ODP.NET. In the table, the term Configuration Support is followed by the types of configuration support (Windows registry, XML file, and so on) that are available for that attribute.

The table describes valid values as well as the default for each attribute.

Note

The default values shown are the values used for an attribute if the registry key does not exist or if it is not configured anywhere.

Table 2-3 Configuration Attributes

Attribute/Setting Name	Description
CheckConStatus	Specifies whether the status of the connection is checked or not before putting the connection back into the connection pool. This registry entry is not created by the installation of ODP.NET. However, the default value 1 is used. Configuration Support: Windows Registry and XML file Valid Values: 1: Check the status of the connection. 0: Do not check the status of the connection. Default: 1

Table 2-3 (Cont.) Configuration Attributes

Attribute/Setting Name	Description
DbNotificationPort	Specifies the port number which ODP.NET listens to, for all notifications sent by the database for change notification, HA, or RLB features. ODP.NET does not throw any errors if an invalid or used port number is specified. The port can also be set to override the Windows registry and XML configuration file by setting the <code>OracleDependency.Port</code> static field. Configuration Support: XML file, and ODP.NET class Valid Values: -1: Open a random unused port to listen to. $n \geq 0$: Listen on port n. Default: -1
DemandOraclePermission	Specifies whether ODP.NET demands <code>OraclePermission</code> from the .NET application that is trying to access the database using ODP.NET. Configuration Support: Windows Registry and XML file Valid Values: 0: Disables demands for <code>OraclePermission</code>. 1: Enables demands for <code>OraclePermission</code> Default: 0
DllPath	Specifies the location where dependent unmanaged Oracle Client binaries load from. Configuration Support: Windows Registry and XML file Valid Values: The path where dependent unmanaged Oracle Client binaries reside. Default: <code>ORACLE_BASE\ORACLE_HOME\bin</code>
DynamicEnlistment	Due to a behavior change with the ODAC 12c Release 3 version of ODP.NET connection string attribute <code>enlist=dynamic</code>, <code>DynamicEnlistment</code> has no operation now.
FetchSize	Specifies the total memory size, in bytes, that ODP.NET allocates to cache the data fetched from a database round-trip. This value can be set on the <code>OracleCommand</code> and the <code>OracleDataReader</code> <code>FetchSize</code> property as well. Configuration Support: Windows Registry, XML file, and ODP.NET class Valid Values: $0 \leq n \leq \text{int.MaxValue}$: n is the size of the cache in bytes. Default: 131072

Table 2-3 (Cont.) Configuration Attributes

Attribute/Setting Name	Description
LegacyEntireLobFetch	Returns either <code>OracleBlob</code> and <code>OracleClob</code> types or <code>OracleBinary</code> and <code>OracleString</code> types from Oracle Database BLOB and CLOB columns. This setting only applies when <code>InitialLobFetchSize</code> is set to -1. Valid Values: 0: Returns <code>OracleBlob</code> and <code>OracleClob</code> 1: Returns <code>OracleBinary</code> and <code>OracleString</code> Default: 0
LegacyTransactionBindingBehavior	Specifies when a database connection detaches from a <code>System.Transactions</code> transaction. By default, connections detach from a transaction only when explicitly unbound as is the case when the connection closes or implicitly unbound when the transaction is disposed. Alternatively, this attribute can be set so that the connection detaches whenever the transaction ends (commits, aborts, or times out), the connection closes, or the transaction is disposed. In ODP.NET 11.2.0.3.20 and earlier releases, the latter was the default behavior. Oracle recommends using the current default behavior. In the earlier default behavior, when the timeout elapses before the transaction completes, the connection unbinds itself from the transaction and all subsequent executions on this connection execute in <code>AutoCommit</code> mode. Any operations prior to the timeout roll back, but operations performed after the timeout commit. In the current default setting, users receive an exception when the transaction times out and additional operations execute on the connection. Configuration Support: Windows Registry and XML file Valid Values: 0: Connections detach from transaction when the connection closes or the transaction is disposed. 1: Connections detach from transaction when the connection closes, the transaction is disposed, or the transaction completes (commits, rolls back, times out). Default: 0
MaxStatementCacheSize	Specifies the maximum number of statements that can be cached when self-tuning is enabled. Configuration Support: Windows Registry and XML file Valid Values: 0 to <code>System.Int32.MaxValue</code>. Default: <code>OPEN_CURSORS</code> setting value of the database

Table 2-3 (Cont.) Configuration Attributes

Attribute/Setting Name	Description
MetaDataXml	Specifies the name of the XML file that customizes the queries to obtain the metadata the ADO.NET 2.0 GetSchema method returns. MetaDataXml can only be set in a configuration file. Configuration Support: XML file only Valid Values: A complete file name for the XML file. Default: <i>none</i>
PerformanceCounters	Enables or disables publishing performance counters for connection pooling. Multiple performance counters can be obtained by adding the valid values. Configuration Support: Windows Registry and XML file Valid Values: 0: Not Enabled 1: Number of sessions being established with Oracle Database every second. 2: Number of sessions being severed from Oracle Database every second. 4: Number of active connections originating from connection pools every second. 8: Number of active connections going back to the connection pool every second. 16: Total number of active connections. 32: Number of inactive connection pools. 64: Total number of connections in use. 128: Total number of connections available for use in all the connection pools. 256: Total number of pooled active and free connections. 512: Number of non-pooled active connections 1024: Number of connections which were garbage-collected implicitly. 2048: Number of connections that will be soon available in the pool. User has closed these connections, but they are currently awaiting actions, such transaction completion, before they can be placed back into the pool as free connections. 4095: All the above Default: 0

Table 2-3 (Cont.) Configuration Attributes

Attribute/Setting Name	Description
SelfTuning	Specifies whether self-tuning is enabled for an ODP.NET application. Configuration Support: Windows Registry, XML file, and Self Tuning connection string attribute Valid Values: 0: Self Tuning is disabled. Used in the registry or XML file. false: Self Tuning is disabled. Used for the Self Tuning connection string attribute. 1: Self Tuning is enabled. Used in the registry or XML file. true: Self Tuning is enabled. Used for the Self Tuning connection string attribute. Default: 1
StatementCacheSize	Specifies the number of cursors or statements to be cached on the database for each connection. This setting corresponds to <i>Statement Cache Size</i> attribute in the connection string. A value greater than zero also enables statement caching. Configuration Support: Windows Registry, XML file, and <i>Statement Cache Size</i> connection string attribute Valid Values: 0 <= <i>n</i> <= the value of OPEN_CURSORS parameter set in <code>init.ora</code> database config file. <i>n</i> is the number to set. Default: 0
StatementCacheWithUdts	Specifies whether or not Oracle UDTs retrieved by executing a SELECT statement are cached along with the statement in the statement cache. This setting affects the memory usage and performance of the application. Configuration Support: Windows Registry and XML file Valid Values: 0: Oracle UDTs are not cached with statements. 1: Oracle UDTs are cached along with statements. Default: 1

Table 2-3 (Cont.) Configuration Attributes

Attribute/Setting Name	Description
ThreadPoolMaxSize	Specifies the default maximum size of worker threads for each available processor in a process. This value may affect the performance of ODP.NET connection creation, command execution timeout, and external procedures (extproc) that use the thread pool. However, unnecessarily increasing thread pool maximum size can also cause performance problems. Configuration Support: Windows Registry and XML file Valid Values: $0 \leq n \leq \text{int.MaxValue}$: Allows ODP.NET to reset thread pool maximum size with the value n. The ODP.NET reset operation may be ignored if the value is invalid. For example, if n is less than the number of available processors of the system. In this case, the result is the same as the value -1. -1: Leave the thread pool max size as is. Default: -1 (this registry entry is not created by default)
TraceFileLocation	Trace file destination directory, for example, D:\traces. Configuration Support: Windows Registry and XML file Default: <windows user temporary folder>\ODP.NET\unmanaged\trace
TraceFileMaxSize	Maximum file size of each trace file. Configuration Support: Windows Registry and XML file Default: 100. Units are in megabytes (that is, 100 MB).
TraceFileName	Specifies the file name to be used for logging trace information. Configuration Support: Windows Registry and XML file Valid Values: Any valid directory location and file name. Default: c:\odpnet2.trc (for .NET Framework 2.0)

Table 2-3 (Cont.) Configuration Attributes

Attribute/Setting Name	Description
TraceLevel	Specifies the level of tracing in ODP.NET. Because tracing all the entry and exit calls for all the objects can be excessive, TraceLevel is provided to limit tracing to certain areas of the provider. Each valid value indicates a possible tracing level. Compounded tracing levels can be obtained by adding the valid values. Configuration Support: Windows Registry and XML file Valid Values: 0: None 1: Entry, exit, and SQL statement information 2: Connection pooling statistics 4: Distributed transactions (enlistment and delistment) 8: User-mode dump creation upon unmanaged exception 16: HA Event Information 32: Load Balancing Information 64: Self Tuning Information 127: All the above Default: 0 Note: ODP.NET does bit-wise checking on the value. When tracing is enabled, logging to the trace file can affect ODP.NET performance. Note: The user-mode dump creation requires dbghelp.dll version 5.1.2600.0 or later.
TraceOption	Specifies whether to log trace information in single or multiple files for different threads. If a single trace file is specified, the file name specified in TraceFileName is used. If the multiple trace files option is requested, a Thread ID is appended to the file name provided to create a trace file for each thread. Configuration Support: Windows Registry and XML file Valid Values: 0: Single trace file 1: Multiple trace files Default: 0

Table 2-3 (Cont.) Configuration Attributes

Attribute/Setting Name	Description
UdtCacheSize	Specifies the size of the object cache for each connection in kilobytes (KB) that ODP.NET uses to retrieve and manipulate Oracle UDTs. Configuration Support: Windows Registry and XML file Valid Values: $0 \leq n \leq 4194303$, n is the number to set. Default: 4096
UDT Mapping	Specifies a mapping between a custom type and an Oracle UDT in the database. The mappings can be specified in configuration files and custom type factories. However, if the mapping is specified in both places, mappings specified in the configuration files takes precedence over mappings specified using custom type factories. Configuration Support: XML file and Custom Type Factory Classes Valid Values: Any valid mapping. Default: <i>none</i>

Windows Registry

Upon installation, ODP.NET creates entries for configuration and tracing within the Windows Registry. Configuration and tracing registry values apply across all ODP.NET applications running in that Oracle client installation. Individual ODP.NET applications can override some of these values by configuring them within the ODP.NET application itself (for example, `FetchSize`). Applications can also use the .NET configuration files to override some of the ODP.NET Windows Registry values.

The ODP.NET registry values are located under `HKEY_LOCAL_MACHINE\Software\Oracle\ODP.NET\version\`. There is one key for .NET Framework 3.5, and one key for .NET Framework 4 and later.

Note

32-bit applications running on an x64-based version of Windows use the registry subkey, `HKEY_LOCAL_MACHINE\Software\Wow6432node` in place of `HKEY_LOCAL_MACHINE\Software`. If such applications use Oracle Data Provider for .NET (32-bit), then the ODP.NET registry values are located under `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Oracle\ODP.NET\version\`.

Configuration File Support

For customers who have numerous applications on a computer that depends on a single version of ODP.NET, the Windows Registry settings for a given version of ODP.NET may not necessarily be applicable for all applications that use that version of ODP.NET. To provide more

granular control, ODP.NET Configuration File Support allows developers to specify ODP.NET configuration settings in an application config, web.config, or a machine.config file.

If a computer does not require granular control beyond configuration settings at the ODP.NET version level, there is no need to specify ODP.NET configuration settings through configuration files.

The following is an example of a web.config file for .NET Framework:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <oracle.dataaccess.client>
    <settings>
      <add name="DllPath" value="C:\oracle\bin"/>
      <add name="FetchSize" value="131072"/>
      <add name="StatementCacheSize" value="10"/>
      <add name="TraceFileName" value="D:\odpnet2.trc"/>
      <add name="TraceLevel" value="63"/>
      <add name="TraceOption" value="1"/>
    </settings>
  </oracle.dataaccess.client>
</configuration>
```

The following is an example of app.config for ODP.NET, Unmanaged Driver using .NET Framework, which sets some additional attributes as well as two UDT type mappings:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <oracle.dataaccess.client>
    <settings>
      <add name="DbNotificationPort" value="-1"/>
      <add name="DllPath" value="C:\app\user\product\21.3.0\client_1\bin"/>
      <add name="DynamicEnlistment" value="0"/>
      <add name="FetchSize" value="131072"/>
      <add name="MetaDataXml" value="CustomMetaData.xml"/>
      <add name="PerformanceCounters" value="4095"/>
      <add name="StatementCacheSize" value="50"/>
      <add name="ThreadPoolMaxSize" value="30"/>
      <add name="TraceFileName" value="D:\odpnet2.trc"/>
      <add name="TraceLevel" value="0"/>
      <add name="TraceOption" value="0"/>
      <add name="Person" value="udtMapping factoryName='PersonFactory, Sample,
        Version=0.0.0.0, Culture=neutral, PublicKeyToken=null' typeName='PERSON'
        schemaName='HR' dataSource='oracle'"/>
      <add name="Student" value="udtMapping factoryName='StudentFactory, Sample,
        Version=0.0.0.0, Culture=neutral, PublicKeyToken=null' typeName='STUDENT'
        schemaName='HR'"/>
    </settings>
  </oracle.dataaccess.client>
</configuration>
```

ODP.NET, Unmanaged Driver now has the option of using the same configuration file format as ODP.NET, Managed Driver. The format simplifies configuration by using a single unified scheme. To utilize this format, the existing unmanaged ODP.NET configuration section should be renamed from `<oracle.dataaccess.client>` to `<oracle.unmanageddataaccess.client>`. The existing unmanaged ODP.NET elements and values are supported within the new section using the same format as with ODP.NET, Managed Driver. To see how to set the elements and values, see ["Oracle Data Provider for .NET, Managed Driver Configuration"](#) for more information.

For example, converting the `FetchSize` element and value from the traditional to the new format would be done as follows:

```
<oracle.dataaccess.client>
  <settings>
    <add name="FetchSize" value="131072" />
  </settings>
</oracle.dataaccess.client>

<oracle.unmanageddataaccess.client>
  <version number="*">
    <settings>
      <setting name="FetchSize" value="131072" />
    </settings>
  </version>
</oracle.unmanageddataaccess.client>
```

The traditional ODP.NET, Unmanaged Driver configuration file format will continue to be supported.

SQL Translation Framework Configuration

Configuring the SQL Translation Profile

The default SQL Translation Profile can be set in the .NET config file, either for all connections across the application, or it is also possible to limit the scope of a profile based on optional `dataSource` and `userId` XML attributes. Please note that these `dataSource` and `userId` XML attributes directly correspond to the `Data Source` and `User Id` attributes in the connection string used to open a database connection.

Note

SQL Translation Profile settings are only supported in the `<oracle.unmanageddataaccess.client>` section. It is not supported in the `<oracle.dataaccess.client>` section nor the `<oracle.manageddataaccess.client>` section.

This would be used for all connections to the `Data Sources` and `User Ids`.

This would be used for all connections to the specified `Data Source`.

This would be used for all connections to the specified `User Id`.

This would be used for all connections to the specified `Data Source` and `User Id`.

It is possible to configure multiple default profile entries which allow configuring default profiles for different `dataSource` and `userId` attributes, but while selecting a profile, the profile with maximum matching attributes will be selected.

In case there are 2 matching entries, one with `dataSource` only and the other with `userId` only then the entry with matching the `userId` would be given priority over the entry with matching `dataSource`.

With the above configuration, if we try to connect with a connection string which has `stf_ds` for `Data Source` and `stf_user` for `User Id` attributes, then both the entries given above will match

and in such cases, we will give priority to the entry with a matching User Id attribute which means profile_user will be selected as the default profile.

Configuring the Error Mapping

Applications can configure the connection related error mapping in their application configuration file. The error mapping can also be scoped based on Data Source name, User Id and the profile name itself.

Here is an example of providing error mapping with all three attributes.

```
<configuration>
  <oracle.unmanageddataaccess.client>
    <version number="*">
      <sqlTranslation>
        <defaultProfiles>
          <defaultProfile dataSource="stf_ds" userId="stf_user" profile=" Profile4"/>
        </defaultProfiles>
        <ErrorMappings>
          <ErrorMapping dataSource="stf_ds" userId="stf_user" profile="Profile4">
            <add oracleErrorNumber="1017" translatedErrorCode="222" />
            <add oracleErrorNumber="1005" translatedErrorCode="888" />
          </ErrorMapping>
        </ErrorMappings>
      </sqlTranslation>
    </version>
  </oracle.unmanageddataaccess.client>
</configuration>
```

Please note that dataSource and userId attributes are optional but can be used to scope the mapping.

It is also possible to provide an error mapping which could be used for all profiles. Here is an example:

```
<ErrorMappings>
  <ErrorMapping profile="*">
    <add oracleErrorNumber="1017" translatedErrorCode="222" />
    <add oracleErrorNumber="1018" translatedErrorCode="888" />
  </ErrorMapping>
</ErrorMappings>
```

Configuring the Default Error Mapping Profile

The default error mapping profile can be configured through the defaultErrorMappingProfile setting. This is to be used to specify the default error mapping profile, especially in scenarios when the default profile is not specified through the .NET configuration file, but specified on the server side. In this case, if connectivity related errors occur, then ODP.NET will be able to properly use error mappings specified in the .NET configuration file for the profile specified by the defaultErrorMappingProfile setting.

Here is an example to configure the default error mapping profile:

```
<sqlTranslation>
  <settings>
    <add name="defaultErrorMappingProfile" value="error_mapping_profile" />
  </settings>
</sqlTranslation>
```

Configuring the SQL Translation Framework Statement Cache Size

Client can configure the number of translated statements that ODP.NET can cache internally to avoid translations, which can be an expensive operation.

Here is an example to configure default error mapping profile:

```
<sqlTranslation>
  <settings>
    <add name="translatedStatementCacheSize" value="50" />
  </settings>
</sqlTranslation>
```

Sample SQL Translation Framework configuration file

Here is a sample configuration file with all possible elements that can be used:

```
<sqlTranslation>
  <settings>
    <add name="translatedStatementCacheSize" value="50" />
    <add name="defaultErrorMappingProfile" value="def_Profile" />
  </settings>
  <defaultProfiles>
    <defaultProfile profile="STF.NO_DS_NO_USERID"/>
    <defaultProfile userId="stf" profile="STF_NO_DS"/>
    <defaultProfile dataSource="stf_inst" profile="STF_NO_USERID"/>
    <defaultProfile dataSource="stf_inst" userId="stf" profile="STF.STF_X"/>
  </defaultProfiles>
  <ErrorMappings>
    <ErrorMapping profile="def_profile">
      <add oracleErrorNumber="1017" translatedErrorCode="444" />
    </ErrorMapping>
    <ErrorMapping dataSource="stf_inst" userId="stf" profile="STF.STF_X">
      <add oracleErrorNumber="1018" translatedErrorCode="88888" />
    </ErrorMapping>
  </ErrorMappings>
</sqlTranslation>
```

Example 2-1 Setting the profile which could be used for all connections

```
<configuration>
  <oracle.unmanageddataaccess.client>
    <version number="*">
      <sqlTranslation>
        <defaultProfiles>
          <defaultProfile profile="Profile1"/>
        </defaultProfiles>
      </sqlTranslation>
    </version>
  </oracle.unmanageddataaccess.client>
</configuration>
```

Example 2-2 Setting the Profile for a Specific Data Source

```
<defaultProfiles>
  <defaultProfile dataSource="stf_ds" profile="Profile2"/>
</defaultProfiles>
```

Example 2-3 Setting the Profile for a Specific User Id

```
<defaultProfiles>
  <defaultProfile userId="stf_user" profile="Profile3"/>
</defaultProfiles>
```

Example 2-4 Setting the Profile for a Specific Data Source and User Id'

```
<defaultProfiles>
  <defaultProfile dataSource="stf_ds" userId="stf_user" profile="Profile4"/>
</defaultProfiles>
```

Example 2-5 Configuring Multiple Default Profile Entries

```
<defaultProfiles>
  <defaultProfile dataSource="stf_ds" profile="profile_ds"/>
  <defaultProfile userId="stf_user" profile="profile_user"/>
</defaultProfiles>
```

Specifying UDT Mappings with Unified Configuration for Unmanaged ODP.NET

As UDT mapping is not currently supported by ODP.NET, Managed Driver, a new section within the <version> section is used to support custom UDT mappings for unmanaged ODP.NET in the unified configuration format. This new section is identified as <udtmappings> and each mapping is identified using a <udtmapping> element. The following attributes may be specified for each udtMapping element:

- typeName (required)
- factoryName (required)
- dataSource (optional)
- schemaName (optional)

These elements retain the same name and meaning as when used with the traditional configuration format.

Example of converting traditional format to unified format:

```
<configuration>
  <oracle.dataaccess.client>
    <settings>
      <add name="Person" value="udtMapping factoryName='PersonFactory, Sample,
Version=0.0.0.0, Culture=neutral, PublicKeyToken=null' typeName='PERSON'
schemaName='SCOTT' dataSource='oracle'" />
    </settings>
  </oracle.dataaccess.client>
</configuration>

<configuration>
  <oracle.unmanageddataaccess.client>
    <udtmappings>
      <udtmapping typename="PERSON" factoryname="PersonFactory, Sample,
Version=0.0.0.0, Culture=neutral, PublicKeyToken=null" schemaname="SCOTT"
datasource="oracle" />
    </udtmappings>
  </oracle.unmanageddataaccess.client>
</configuration>
```

① See Also[Oracle User-Defined Types \(UDTs\) and .NET Custom Types](#)

Oracle Data Provider for .NET, Managed Driver Configuration

ODP.NET, Managed Driver supports .NET configuration file-based settings in `machine.config`, `app.config`, and `web.config`. It does not support Windows registry based configuration. ODP.NET, Managed Driver settings in .NET configuration files are similar to ODP.NET, Unmanaged Driver settings to make porting easier.

The ODP.NET, Managed Driver configuration file section name is `<oracle.manageddataaccess.client>`. The `<oracle.manageddataaccess.client>` settings and values are also supported in unmanaged ODP.NET configuration file: `<oracle.unmanageddataaccess.client>`. While this documentation section discusses managed ODP.NET configuration, it is also applicable to `<oracle.unmanageddataaccess.client>`. The `<oracle.unmanageddataaccess.client>` share numerous common settings with `<oracle.manageddataaccess.client>`. Differences between the two are noted on this page. The `<oracle.unmanageddataaccess.client>` settings not available in managed ODP.NET are documented in "[Oracle Data Provider for .NET, Unmanaged Driver Configuration](#)". A typical .NET config that uses ODP.NET, Managed Driver has some or all of the following subsections nested within a `<version>` subsection under `<oracle.manageddataaccess.client>` section. Note the tag names are case sensitive, while the attribute names are case insensitive.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <oracle.manageddataaccess.client>
    <version number="*">
      <dataSources>
        ...
      </dataSources>
      <settings>
        ...
      </settings>
      <LDAPsettings>
        ...
      </LDAPsettings>
      <implicitRefCursor>
        ...
      </implicitRefCursor>
      <edmMappings>
        ...
      </edmMappings>
    </version>
    <version number="4.121.2.0">
      <dataSources>
        ...
      </dataSources>
      <settings>
        ...
      </settings>
    </version>
  </oracle.manageddataaccess.client>
</configuration>
```

```

    ...
  </settings>
  <LDAPsettings>
    ...
    ...
  </LDAPsettings>
  <implicitRefCursor>
    ...
    ...
  </implicitRefCursor>
  <edmMappings>
    ...
    ...
  <edmMappings>
</version>
</oracle.manageddataaccess.client>
</configuration>

```

The ODP.NET, Managed Driver configuration and settings are described in the following sections. Many of the attributes are the same as ODP.NET, Unmanaged Driver. See [Table 2-3](#) for detailed attribute descriptions.

version Section

All the information required by an application should be grouped under the version subsections. Each `<version number="X">` section contains parameters applicable for version X of the ODP.NET, Managed Driver. For example, `<version number="4.121.2.0">` section parameters will be applicable only for those applications using ODP.NET, Managed Driver assembly 4.121.2.0.

Apart from version specific sections, there can also be a generic section `<version number="*">`. This section's parameters are applicable for all ODP.NET, Managed Driver versions. Parameters in the version specific section take precedence over the parameters of the generic section. The following is an example of a version section:

```

<oracle.manageddataaccess.client>
  <version number="*">
    <settings>
      <setting name="TraceOption" value="1"/>
      <setting name="PerformanceCounters" value="0" />
    </settings>
  </version>
  <version number="4.121.2.0">
    <settings>
      <setting name="PerformanceCounters" value="4095" />
    </settings>
  </version>
</oracle.manageddataaccess.client>

```

An application referencing ODP.NET, Managed Driver 4.121.2.0 has the following values set:

- TraceOption = 1
- PerformanceCounters= 4095

dataSources Section

This section can appear only under a `<version>` section. The mapping between the different data source aliases and corresponding data descriptors should appear in this section. This section is supported by managed ODP.NET only. The following is an example.

```
<dataSources>
  <dataSource alias="inst1" descriptor="(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=sales-
server).....))"/>
  <dataSource alias="inst2" descriptor="(DESCRIPTION= .....))"/>
</dataSources>
```

Note

The data source connection string attribute can alternatively be set to a full descriptor or Easy Connect syntax rather than a data source alias.

Requirements for connecting to a local database *without* specifying "data source" connection string attribute:

- The listener must be up and running.
- ORACLE_SID environment variable must be set appropriately.

Note

When data source connection string attribute is not specified, protocol defaults to 'tcp' and port defaults to '1521'.

The ODP.NET managed driver reads and caches all the alias entries from the app.config, web.config, machine.config, and from a tnsnames.ora file that is found at application start-up time. However, aliases that are defined in LDAP servers are resolved and cached on demand. This means for each unique alias that is used by the application, an alias resolution query is executed against an LDAP server and the full descriptor associated with the alias will be cached once it is fetched.

For developers that need to change or add alias settings while developing applications, one may consider using `OracleDataSourceEnumerator.GetDataSources()` rather than restarting the application. Invoking this method will first wipe out existing cache entries that were read from the tnsnames.ora file and all aliases obtained from the LDAP Server. Then, the tnsnames.ora is re-parsed and all its entries will be cached again. Please note that the app.config, web.config, and machine.config entries are read only once at application start-up time and thus their contents are maintained and not re-parsed even if `OracleDataSourceEnumerator.GetDataSources()` is invoked.

The `OracleDataSourceEnumerator.GetDataSources()` method invocation has an impact on the connection pool. This is because a connection pool, which is created for each unique connection string, will cache the resolved full descriptor information after the first connection is created for a given connection pool. After that, the connection pool uses the cached full descriptor information for all subsequent connection creations. Thus, for applications that have their tnsnames.ora or LDAP entries modified during the execution of an application where an alias points to a different database than before, one should call the `OracleDataSourceEnumerator.GetDataSources()` method to remove old cached entries. This should be followed by the invocation of the `ClearPool(OracleConnection)` instance method or the `ClearAllPools()` static method to remove existing connections and also have it obtain a new full descriptor value that was read by the invocation of `OracleDataSourceEnumerator.GetDataSources()`. Following this scheme will assure that *all* the connections in the connection pool uses the new full descriptor that is now associated with the alias and all connections in a connection pool is established to the same database.

The following keywords are supported within the descriptor setting:

- ADDRESS
- ADDRESS_LIST (Note: only failover supported)
Oracle recommends using SCAN listener and Runtime Load Balancing to balance the load when connecting to an Oracle RAC database.
- DESCRIPTION
- DESCRIPTION_LIST (Note: Failover supported; Address_list load balancing not supported)
- HOST (Note: <hostname>, <IPv6 literal>, and <IPv4 literal> are supported)
- IP (Note: "loopback" is supported)
- PROTOCOL (Note: tcp and tcps are supported. In addition, WS and WSS are also supported.)
- RETRY_COUNT: Specifies the number of times an ADDRESS list is traversed before the connection attempt is terminated and times out.
- RETRY_DELAY: Specifies the delay in seconds between subsequent retries for a connection.

RETRY_COUNT and RETRY_DELAY are timeout parameters that work at the Oracle networking layer. They are intended to be used in conjunction with each other. When these parameters are used, ODP.NET will attempt to connect with one of the addresses in the ADDRESS_LIST. If there is no successful connection after going through the entire list, then ODP.NET will wait for the number of seconds specified by RETRY_DELAY before traversing the address list again. It will repeat this behavior until there is a successful connection or the RETRY_COUNT limit is hit.

For example, let's assume RETRY_COUNT=2 and RETRY_DELAY=4 when ODP.NET cannot successfully connect to any address on the list. The following sequence is triggered:

1. Traverse address list to connect.
2. Wait four seconds for next set of retry attempts.
3. Traverse the address list again. This is first set of retry attempts.
4. Wait four seconds for the next retry attempt.
5. Traverse the address list again. This is the second and final set of retry attempts.
6. Timeout

These timeout parameters function independent of ODP.NET application timeouts, such as the Connection Timeout connection string attribute. If the Connection Timeout is set to a low value, then it may trigger before the RETRY_COUNT and RETRY_DELAY attempt cycle completes.

- Session Data Unit size supports from 256 to 2097152 in bytes.
- SECURITY: SSL_VERSION (Note: overrides sqlnet.ora:ssl_version)
- TRANSPORT_CONNECT_TIMEOUT (Note: overrides tcp.connect_timeout)

Note

- SSL is now supported via method MCS and FILE.
- Both Kerberos5 and NTS authentication are supported. RADIUS is not supported.
- Only Net Services, Easy Connect naming, and LDAP (namely, Active Directory and Oracle Internet Directory) are supported.
- No bequeath (beq) support. Default address is instead TCP loopback with port 1521 and Oracle service name from environment (ORACLE_SID)

See Also

Oracle Database Net Services Reference for a detailed description of the attributes.

Though managed ODP.NET does not support TNS descriptor based load balancing, it does support failover through both an ADDRESS_LIST and DESCRIPTION_LIST.

Note that you need not specify either the LOAD_BALANCE or the FAILOVER directive, because only failover is supported. The directives are ignored.

The following examples demonstrate TNS descriptors utilizing failover:

```
(DESCRIPTION=
  (ADDRESS_LIST=
    (ADDRESS=(PROTOCOL=tcp)(HOST=host1)(PORT=1630))
    (ADDRESS=(PROTOCOL=tcp)(HOST=host2)(PORT=1630))
    (ADDRESS=(PROTOCOL=tcp)(HOST=host3)(PORT=1521)))
  (CONNECT_DATA=(SERVICE_NAME=Sales.us.example.com)))

(DESCRIPTION_LIST=
  (DESCRIPTION=
    (ADDRESS_LIST=
      (ADDRESS=(PROTOCOL=tcp)(HOST=sales1a-svr)(PORT=1521))
      (ADDRESS=(PROTOCOL=tcp)(HOST=sales1b-svr)(PORT=1521)))
    (CONNECT_DATA=(SERVICE_NAME=sales1.example.com)))
  (DESCRIPTION=
    (ADDRESS_LIST=
      (ADDRESS=(PROTOCOL=tcp)(HOST=sales2a-svr)(PORT=1521))
      (ADDRESS=(PROTOCOL=tcp)(HOST=sales2b-svr)(PORT=1521)))
    (CONNECT_DATA=(SERVICE_NAME=sales2.us.example.com))))
```

settings section

This section can appear only under a <version> section. Any ODP.NET, Managed Driver specific settings should appear in this section. The following is an example of a settings section:

```
<settings>
  <setting name="TraceLevel" value="7" />
  <setting name="TraceOption" value="1"/>
  <setting name="TNS_ADMIN" value="C:\oracle\work"/>
</settings>
```


A new default behavior has been introduced for ODP.NET Release 12.1.0.2 and later when `InitialLobFetchSize` is set to -1. The new default value is `LegacyEntireLobFetch = 0`. To use the old behavior, set `LegacyEntireLobFetch = 1` in the ODP.NET configuration. Refer to ["Setting InitialLONGFetchSize to -1"](#) for more information.

ODP.NET, Managed Driver configuration settings that are supported:

- `BindByName`
- `CPVersion`: Determines whether ODP.NET, Unmanaged Driver uses the traditional connection pool implementation (`Default=1.0`) or the more modern connection pool implementation with better high availability support (`2.0`). In addition, the more modern connection pool (`2.0`) is required if the application uses Database Resident Connection Pooling (DRCP), or hosts multiple pluggable databases or editions in the same pool. Possible values: `1.0` or `2.0`.

ODP.NET, Managed Driver only uses the more modern connection pool implementation. This setting can only be used for ODP.NET, Unmanaged Driver for .NET Framework 4 and higher. It cannot be used for earlier versions of unmanaged ODP.NET, which will always use the traditional connection pool implementation.

If `CPVersion` is set explicitly, then that setting will be used across processes for all connection strings.

If `CPVersion` is left unmodified (or contains an invalid value) and an application's first unmanaged ODP.NET established connection for a given connection string uses DRCP, pluggable databases, or editions by modifying one these features' properties or configuration settings, then `CPVersion` will be implicitly modified to use the newer connection pool implementation (`2.0`).

If a `CPVersion 1.0` pool attempts to use DRCP, pluggable database, or edition settings, then ODP.NET will raise an exception when the application attempts to open a connection.

When `CPVersion 2.0` is used explicitly or implicitly and the "proxy user id" is not specified in the connection string, then a proxy connection will not be created, even if "proxy password" is supplied.

When `CPVersion 2.0` is used by ODP.NET, Unmanaged Driver, the connection pool behavior will be the same as the more modern connection pool as implementation that ODP.NET, Managed Driver uses.

- `DbNotificationPort`
- `DemandOraclePermission`
- `Disable_0ob`: Interrupts database query execution via either TCP/IP urgent data or normal TCP/IP data, called out of band data (default) or in band data, respectively. (`Default=off`).

All Oracle database clients support interrupting database query execution, such as through an ODP.NET command timeout. Windows-based database servers only support in band breaks, whereas all other (predominantly UNIX-based) database servers can support out of band (OOB) or in band breaks. ODP.NET, Managed Driver uses OOB breaks by default with database servers that support it. For certain network topologies, the routers or firewalls involved in the route to the database may have been configured to drop urgent data or in band the data. If the routers or firewalls can not be changed to handle urgent data appropriately, then the ODP.NET, Managed Driver can be configured to utilize in band breaks by setting the .NET configuration parameter `Disable_0ob` to on.

- `DllPath`: (Unmanaged ODP.NET only) Specifies the directory location dependent unmanaged Oracle Client binaries are loaded from.

See [Search Order for Unmanaged DLLs](#) for more information.

- **DRCPConnectionClass**: Specifies a logical name that identifies the DRCP connection pool that the ODP.NET connection will use. It will be used as a default if the **DRCPConnectionClass** property on the **OracleConnection** object is not set. It will be ignored for non-DRCP connections.

Valid Values: the connection class name is string at max 1024 characters. The default is null.

- **FetchSize**
- **GetDecimalRetainTrailingZeros**: (Managed ODP.NET only) Specifies whether to retain trailing zeros from an Oracle NUMBER in a .NET Decimal.

See [OracleConfiguration GetDecimalRetainTrailingZeros](#) for more information.

- **LDAP_ADMIN**: Specifies the `ldap.ora` location. The **LDAP_ADMIN** setting works in conjunction with the **TNS_ADMIN** setting to set `ldap.ora` search order.

See [Oracle Client Configuration File Settings](#) for `ldap.ora` search order.

- **LegacyEntireLOBFetch**
- **MaxStatementCacheSize**
- **MetaDataXml**
- **NAMES.DIRECTORY_PATH**: The default search order is **TNSNAMES** and **EZCONNECT**. **TNSNAMES**, **LDAP**, and **EZCONNECT** are the only name resolution methods supported, but their order of precedence can be modified.
- **NAMES.LDAP_AUTHENTICATE_BIND**
- **NAMES.LDAP_AUTHENTICATE_BIND_METHOD**
- **NAMES.LDAP_CONN_TIMEOUT**
- **NODELAY**
- **ORA_DEBUG_JDWP**: Allows Oracle PL/SQL Debugger and database to connect automatically without application code changes. Value is set as `host=<IP_address or host_name>;port=<debugging port number>`. Ex. `host=localhost;port=1234`
- **OCI_COMPARTMENT**
- **OCI_CONFIG_FILE**
- **OCI_DATABASE**
- **OCI_PROFILE**
- **OpenTelemetryTracing**: Enable (true) or disable (false) ODP.NET OpenTelemetry tracing instrumentation. Default is true. ODP.NET will only read this property during app startup. It cannot be used to change tracing during runtime.
- **ORACLE_SID**
- **PASSWORD_AUTH**: Possible values are **OCI_TOKEN** and **PASSWORD_VERIFIER** (default).
- **PerformanceCounters**
- **Pipelining**: (Managed ODP.NET only) Specifies whether to enable asynchronous execution on the database server side.
- **RECEIVE_BUF_SIZE**: Sets TCP `SO_RECVBUF`, the total buffer space associated with the local side of a TCP socket
- **RowsToFetchPerRoundTrip**
- **SelfTuning**

- `SEND_BUF_SIZE`: Sets TCP `SO_SENDBUF`, the total buffer space associated with the local side of a TCP socket
- `ServiceRelocationConnectionTimeout`
In seconds. (Default = 90).

Whenever a database service becomes unavailable, such as due to a service being relocated, an application can encounter numerous connectivity errors during this time. To avoid unnecessary connection attempts to an unavailable service which will result in an error, ODP.NET blocks any connection attempts until the service is up or until this property's specified time limit expires from the time when the service DOWN event was received, whichever comes first. Once the specified time elapses, all the connection attempts to the specific service which is known to be down will no longer be blocked. Those requests will be sent to the server. `ServiceRelocationConnectionTimeout` is only operational in conjunction with Oracle Fast Connection Failover (`HA Events = true`). Once Fast Connection Failover is enabled for the .NET application, Service Relocation Connection Timeout is automatically enabled. It will use its default value if no `ServiceRelocationConnectionTimeout` value has been explicitly set. It works with planned and unplanned outages.

When connecting to Oracle Data Guard in Oracle Database 12c Release 2 or later, database administrators can set their own timeout value, `drain_timeout`, to indicate the number of seconds allowed for resource draining to be completed. This setting can be optionally used by the ODP.NET `ServiceRelocationConnectionTimeout` value as the intent behind `drain_timeout` and `ServiceRelocationConnectionTimeout` are the same.

If `ServiceRelocationConnectionTimeout` and `drain_timeout` are not set, then the default `ServiceRelocationConnectionTimeout` value is used (90 seconds). If only `drain_timeout` is set, then that value will be used for the service relocation connection timeout. If only `drain_timeout` is set and is set to 0, then also default `ServiceRelocationConnectionTimeout` value is used (90 seconds). If both are set, then the value of `ServiceRelocationConnectionTimeout` will override the value of `drain_timeout`.

`ServiceRelocationConnectionTimeout` itself can be set based upon the value of `drain_timeout`. For example, if ODP.NET administrators desire the service relocation connection timeout to be `drain_timeout` plus 50 seconds, then they would set `ServiceRelocationConnectionTimeout` in the following manner:

```
<setting name="ServiceRelocationConnectionTimeout" value="drain_timeout + 50"/>
```

Service Relocation Connection Timeout is not supported in pools that connect to more than one pluggable database.

- `SQLNET.AUTHENTICATION_SERVICES`: Supported values are ALL, Kerberos5, NTS, TCPS, or NONE.

Managed ODP.NET supports NTS, Kerberos5, and TCPS external authentication methods. This setting should be set based on the desired database authentication method. If internal database authentication is desired, then the setting should be set to NONE. Default value is NONE. The value(s) must be enclosed in parentheses. Example settings made in `sqlnet.ora` are:

```
SQLNET.AUTHENTICATION_SERVICES = (TCPS)
SQLNET.AUTHENTICATION_SERVICES = (NTS)
SQLNET.AUTHENTICATION_SERVICES = (Kerberos5, NTS)
SQLNET.AUTHENTICATION_SERVICES = (NONE)
```

Note

The NTS external authentication methodology is only supported on a Windows-based client and server.

SQLNET.AUTHENTICATION_SERVICES default values are different for managed ODP.NET and unmanaged ODP.NET. Managed ODP.NET defaults to NONE and unmanaged defaults to ALL when no value is set.

- SQLNET.CLOUD_USER
- SQLNET.CRYPTO_CHECKSUM_CLIENT: Specifies the desired data integrity behavior when this client connects to a server. Supported values are accepted, rejected, requested, or required. Default = accepted.
- SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT: Specifies the data integrity algorithms that this client uses. Supported values are SHA512, SHA384, SHA256, and SHA1.
- SQLNET.ENCRYPTION_CLIENT = Negotiates whether to turn on encryption. Supported values are accepted, rejected, requested, or required.
- SQLNET.ENCRYPTION_TYPES_CLIENT = Encryption algorithm(s) to use.

The following table lists the valid encryption algorithms for ODP.NET, Managed Driver.

Table 2-4 Encryption Algorithms for ODP.NET, Managed Driver

Algorithm Name	Legal Value
AES 128-bit key	AES128
AES 192-bit key	AES192
AES 256-bit key	AES256
2-key 3DES	3DES112
3-key 3DES	3DES168

For more information on data encryption settings, refer to the *Oracle Database Security Guide*.

- SQLNET.EXPIRE_TIME: Specifies a time interval to send a new set of probes to verify the connection remains active. This setting is equivalent to the `OracleConnection.KeepAliveTime` property. `OracleConnection.KeepAliveInterval` is fixed at 6 seconds for SQLNET.EXPIRE_TIME. Default is 0 (off). By default the value is specified in minutes. However the value can be specified in seconds (sec) and milliseconds (ms), such as the following example:

```
<setting name=" SQLNET.EXPIRE_TIME" value="30 sec" />
```

- SQLNET.URI
- SQLNET.WALLET_OVERRIDE
- StatementCacheSize
- SSL_SERVER_DN_MATCH: To enforce the distinguished name (DN) for the database server matches its service name. (Default=no).

If you enforce the match verification, then SSL/TLS ensures that the certificate is from the server. If you select to not enforce the match verification, then SSL/TLS performs the

check but allows the connection, regardless if there is a match. Not enforcing the match allows the server to potentially fake its identify.

Supported values: yes | on | true to enforce a match.

Supported values: no | off | false to not enforce a match.

SSL_SERVER_DN_MATCH is often used together with SSL_SERVER_CERT_DN.

SSL_SERVER_CERT_DN specifies the distinguished name (DN) of the database server. It can be set in the connect descriptor.

```
net_service_name=
  (DESCRIPTION=
    (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-svr)(PORT=1521))
    (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-svr)(PORT=1521))
    (CONNECT_DATA=
      (SERVICE_NAME=sales.us.acme.com))
      (SECURITY=
        (SSL_SERVER_CERT_DN="cn=sales,cn=OracleContext,dc=us,dc=acme,dc=com")))
```

The client uses this information to obtain the list of DNs it expects for each of the servers, enforcing the database server DN to match its service name. Use this parameter with SSL_SERVER_DN_MATCH to enable server DN matching.

- **SSL_VERSION:** Sets the version of the SSL/TLS connection. By default, all supported versions are enabled, in the order 1.2 and 1.3.

The client and server negotiate to the highest version among the common conversions specified in their configurations. The versions from lowest to highest are: 1.2 and 1.3.

To specify more than one version, enclose the version numbers between parentheses and separate the values with a comma. For example, (1.2,1.3).

- **TimeZoneFileLocation:** Specifies the directory location of the time zone file ODP.NET should use for its time zone settings.
- **TNS_ADMIN:** Location where either one or more of tnsnames.ora, ldap.ora, and sqlnet.ora are located. Locations can consist of either absolute or relative directory paths.
- **TOKEN_AUTH:** This attribute specifies the access token authentication type. Possible values are OCI_API_KEY, OCI_INSTANCE_PRINCIPAL, OCI_DELEGATION_TOKEN, OCI_INTERACTIVE, OCI_DEFAULT, OCI_TOKEN, OAUTH, or DISABLED.
- **TOKEN_LOCATION:** This attribute is the file-based token location. The value can be a directory where a file named "token" is or it can be the file's full path specification.
- **TraceFileLocation:** Trace file destination directory, for example, D:\traces. The default TraceFileLocation is <Windows user temporary folder>\ODP.NET\managed\trace.
- **TraceFileMaxSize:** Maximum file size of each trace file. Default is 100 MB (100,000,000). The default value is 100. The unit of measure is megabytes (i.e. 100 MB).
- **TraceLevel:** 1 = public APIs; 2 = private APIs; 4 = network APIs/data; 8 = disables writing SQL statements and network packet contents. These values can be 0Red. To enable everything, set TraceLevel to 7. Errors will always be traced.
- **TraceOption**
- **TCP.CONNECT_TIMEOUT**
- **UseClientInitiatedCQN**
- **WALLET_LOCATION:** Microsoft Certificate Store (MCS) and file system wallets are supported.

① See Also*Oracle Database Net Services Reference*

LDAPsettings section

This section can appear only under a <version> section. Any ODP.NET, Managed Driver specific LDAP settings should appear in this section. This section is supported by managed ODP.NET only. The following is an example of a <LDAPsetting> subsection under the <LDAPsettings> section:

```
<LDAPsettings>
  <LDAPsetting name="DIRECTORY_SERVER_TYPE" value="AD" />
  <LDAPsetting name="DEFAULT_ADMIN_CONTEXT" value="dc=Oracle,dc=com"/>
</LDAPsettings>
```

Lightweight Directory Access Protocol

ODP.NET, Managed Driver supports TNS alias resolution through a LDAP server/service, specifically Microsoft Active Directory and Oracle Internet Directory (OID). TNS alias resolution occurs when using the LDAPsettings section or ldap.ora file settings. The LDAPsettings section settings take precedence over ldap.ora settings.

For Active Directory, only the DIRECTORY_SERVER_TYPE and DEFAULT_ADMIN_CONTEXT parameters are required in ldap.ora. When the DIRECTORY_SERVERS parameter is missing or has no value, the default LDAP server for the current domain will be used.

For OID, all ldap.ora parameters must be set with valid values to complete configuration.

ODP.NET, Managed Driver and ODP.NET, Unmanaged Driver support the same level of security when using LDAP for name resolution.

Table 2-5 Microsoft Active Directory: Encryption Types and Authentication Credentials For Connecting and Binding

No Encryption	SSL Encryption
Anonymous authentication	Anonymous authentication
Domain User authentication	Domain User authentication

Table 2-6 Oracle Internet Directory: Encryption Types and Authentication Credentials For Connecting and Binding

No Encryption	SSL Encryption
Anonymous authentication	Anonymous authentication
-	Wallet based authentication
	<i>Note: Wallet based authentication for Oracle Internet Directory is not supported for this release</i>

See Also

- *Oracle Database Net Services Reference* for more information on Directory Usage Parameters.
- *Oracle Database Net Services Administrator's Guide* for more information on Managing Network Address.

implicitRefCursor section

This section can appear only under a <version> section. Any information about REF CURSOR parameters that need to be bound implicitly should appear in this section. The following is an example of an <implicitRefCursor> section:

```
<implicitRefCursor>
  <storedProcedure schema="USERREFCUR" name="TestProc1">
    <refCursor name="Param3">
      <bindInfo mode="Output"/>
      <metadata columnOrdinal="0" columnName="DEPTNO" baseColumnName="DEPTNO"
baseSchemaName="USERREFCUR" baseTableName="DEPT" nativeDataType="number"
providerType="Int32" dataType="System.Int16" columnSize="2" allowDBNull="true" />
      <metadata columnOrdinal="1" columnName="DNAME" baseColumnName="DNAME"
baseSchemaName="USERREFCUR" baseTableName="DEPT" nativeDataType="varchar2"
providerDbType="String" columnSize="30" />
    </refCursor>
    <refCursor name="param2">
      <bindInfo mode="Output"/>
      <metadata columnOrdinal="0" columnName="EMPNO" baseColumnName="EMPNO"
baseSchemaName="USERREFCUR" baseTableName="EMP" nativeDataType="number"
providerType="Int32" dataType="System.Int16" columnSize="4" allowDBNull="false" />
    </refCursor>
  </storedProcedure>

  <!--Next stored procedure information-->
  <storedProcedure name="TestProc2">
    ...
  </storedProcedure>
</implicitRefCursor>
```

distributedTransaction section

This section can appear only under a <version> section. Any information about distributed transactions should appear in this section. The following is an example of a distributedTransaction section:

```
<distributedTransaction>
  <setting name="OMTSRECO_IP_ADDRESS" value="my-pc" />
  <setting name="OMTSRECO_PORT" value="2040" />
  <setting name="ORAMTS_SESS_TXNTIMETOLIVE" value="240" />
</distributedTransaction>
```

- OMTSRECO_IP_ADDRESS: Specifies the machine name (or IP address) that the OraMTS Recovery service will be running on to resolve database in-doubt transactions. The default is the local machine name.

- OMTSRECO_PORT: Specifies the port that the OraMTS Recovery service will be listening on to resolve database in-doubt transactions. The default is 2030.
- ORAMTS_SESS_TXNTIMETOLIVE : Specifies the time in seconds that the transaction can remain inactive after it has been detached or delisted from the database. Once this time expires, the transaction is automatically terminated by the provider. The default is 120 seconds.
- UseOraMTSManaged: When set to true and using .NET Framework 4.5.2 or higher, ODP.NET uses managed code for distributed transactions. If set to false, ODP.NET uses Oracle Services for Microsoft Transaction Server to support distributed transactions. Boolean (Default = false) for ODP.NET, Unmanaged Driver only.

connectionPools section

This section can appear only under a <version> section. This section allows setting a string identifier for each set of monitored connection counters. Refer to [Connection Performance Counters](#) documentation section for more details.

edmMappings section

This section can appear only under a <version> section. Any information related to EDM mappings should appear in this section. Refer to [Oracle Number Default Data Type Mapping and Customization](#) for more examples on edmMappings section.

onsConfig section

Oracle Notification Service (ONS) can be configured using either local or remote configuration using the <onsConfig> section. The <onsConfig> section is available only for managed ODP.NET. Remote configuration is the preferred configuration for standalone client applications. For releases earlier than Oracle Database 12c, this section is mandatory for ODP.NET to receive ONS notifications. With Oracle Database 12c and later, this section is optional and the information about the ONS daemons is received from the server itself. However, ODP.NET will also listen for events from any <host:port> pairs that is provided by the user in this section in addition to the <host:port> pairs received from the server.

For local configuration, please ensure that ONS is configured and available on the node where ODP.NET is running, so that ODP.NET can receive events directly from the local ONS daemon. The following is a sample format for the local configuration:

```
<onsConfig configFile="C:\temp\test.config" mode="local">
</onsConfig>
```

Note

The configFile specified in .NET config should contain the same localport and remoteport values as specified in the ons.config used by the local ONS daemon. This will enable the application to receive events from the local ONS daemon.

Remote configuration is used in scenarios where the application directly receives ONS events from the ONS daemons running on remote machines. One of the advantages of this configuration is that no ONS daemon is needed on the client end and, therefore, there is no need to manage this process.

The following is a sample format for remote configuration:

```
<onsConfig mode="remote">
  <ons database="db1">
    <add name="nodeList" value="racnode1:4100, racnode2:4200" />
  </ons>
  <ons database="db2">
    <add name="nodeList" value=" racnode3:4100, racnode4:4200" />
  </ons>
</onsConfig>
```

In case of remote configuration, the application has to specify the <host>:<port> values for every potential database that it can connect to. The <host>:<port> value pairs represent the ports on the different Oracle RAC nodes where the ONS daemons are talking to their remote clients.

① See Also

[Client Side ONS Daemon Configuration](#) for information about client side ONS daemon configuration

ONS TCPS and Wallets

ODP.NET enables ONS communications to occur over TCP/IP with SSL/TLS (TCPS), which is more secure than just TCP/IP. As TCPS requires using a wallet for storing keys and certificates, ODP.NET can use one wallet for both ONS and ODP.NET connections or have separate wallets for each.

TCPS provides more secure ONS communication, gives administrators flexibility in how to configure their wallets, and enables cloud database connections.

oracle.manageddataaccess.client configuration section sample

```
<oracle.manageddataaccess.client>
  <version number="*">
    <onsConfig mode="remote">
      <settings>
        <setting name="Protocol" value="TCPS" />
        <setting name="WALLET_LOCATION" value="D:\user\ONS_SSLWallet" />
      </settings>
      <ons database="acdb183">
        <add name="nodeList" value="slcai611:6205,slcai610:6205,slcai612:6205" />
      </ons>
    </onsConfig>
  </version>
</oracle.manageddataaccess.client>
```

The protocol and wallet location properties are often used together.

If an ONS wallet location is set, ODP.NET will use the wallet for ONS to connect via TCP/IP with SSL/TLS regardless of the protocol setting.

If the protocol is set to "TCPS", ODP.NET first tries to use the wallet in the ONS configuration wallet location to connect via TCP/IP with SSL/TLS. If no wallet is present, ODP.NET next tries to use the wallet from the database wallet location. If no wallet is found, the connection fails.

Administrators can choose between having the ONS connection use its own independent wallet or share the database's wallet for ease of use.

If the protocol is set to "TCP" and no ONS wallet location is set, ODP.NET ONS will connect with TCP/IP.

ODP.NET ONS TCPS works with any Oracle Database version that supports ONS TCPS on the server side.

Client Side ONS Daemon Configuration

ONS configuration is controlled by the ONS configuration file, `ORACLE_HOME/opmn/conf/ons.config`. This file tells the ONS daemon how it should behave. The `SRVCTL` utility can be used to start and stop the ONS daemon. It is installed on each node by default during server install.

Configuration information within `ons.config` is defined in simple name and value pairs. An example of `ONS.config` is given below

```
# This is an example ons.config file
#
# The first three values are required
localport=4100
remoteport=4200
nodes=racnode1.example.com:4200,racnode2.example.com:4200
```

Some parameters in the `ons.config` file are required and some are optional. [Table 2-7](#) lists the required ONS configuration parameters and [Table 2-8](#) lists the optional ONS configuration parameters.

Table 2-7 Required ONS Configuration Parameters

Parameter	Explanation
localport	The port that ONS binds to on the local host interface to talk to local clients. For example, <code>localport=4100</code>
remoteport	The port that ONS binds to on all interfaces for talking to other ONS daemons. For example, <code>remoteport=4200</code>
nodes	A list of other ONS daemons to talk to. Node values are given as a comma-delimited list of either host names or IP addresses plus ports. The port value that is given is the remote port that each ONS instance is listening on. In order to maintain an identical file on all nodes, the <code>host:port</code> of the current ONS node can also be listed in the nodes list. It will be ignored when reading the list. For example, <code>nodes=myhost.example.com:4200,123.123.123.123:4200</code> The nodes listed in the nodes line correspond to the individual nodes in the Oracle RAC instance. Listing the nodes ensures that the middle-tier node can communicate with the Oracle RAC nodes. At least one middle-tier node and one node in the Oracle RAC instance must be configured to see one another. As long as one node on each side is aware of the other, all nodes are visible. You need not list every single cluster and middle-tier node in the ONS configuration file of each Oracle RAC node. In particular, if one ONS configuration file cluster node is aware of the middle tier, then all nodes in the cluster are aware of it.

Table 2-8 Optional ONS Configuration Parameters

Parameter	Description
loglevel	The level of messages that should be logged by ONS. This value is an integer that ranges from 1, which indicates least messages logged, to 9, which indicates most messages logged. The default value is 3. For example, loglevel=3
logfile	A log file that ONS should use for logging messages. The default value for log file is \$ORACLE_HOME/opmn/logs/ons.log. For example, logfile=C:\app\user\product\12.1.0\opmn\logs\myons.log
walletfile	The wallet file used by the Oracle Secure Sockets Layer (SSL) to store SSL certificates. If a wallet file is specified to ONS, then it uses SSL when communicating with other ONS instances and require SSL certificate authentication from all ONS instances that try to connect to it. This means that if you want to turn on SSL for one ONS instance, then you must turn it on for all instances that are connected. This value should point to the directory where your ewallet.p12 file is located. For example, walletfile=C:\app\user\product\12.1.0\opmn\conf\ssl.wlt\default
useocr	The value, reserved for use on the server-side, to indicate ONS whether it should store all Oracle RAC nodes and port numbers in Oracle Cluster Registry (OCR) instead of the ONS configuration file or not. A value of useocr=on is used to store all Oracle RAC nodes and port numbers in Oracle Cluster Registry (OCR). Do not use this option on the client-side.

The ons.config file allows blank lines and comments on lines that begin with the number sign (#).

① See Also

Oracle Real Application Clusters Administration and Deployment Guide for more information about the SRVCTL utility.

Relative Windows Path and Windows Environment Variable Configuration Settings

The following managed ODP.NET configuration settings support relative Windows path and environment variables:

- TraceFileLocation
- WALLET_LOCATION

File locations for the above config parameters can now be set using relative Windows paths. The "." notation informs ODP.NET to use the current working directory. Sub-directories can be added by appending them. For example, .\mydir refers to the sub-directory mydir in the current working directory. To navigate to a parent directory, use the ".." notation.

For web applications, the current working directory is the application directory. For Windows applications, the .EXE location is the current working directory.

Windows paths can also be set using Windows environment variable names within "%" characters.

For example, %tns_admin%, c:\%dir%\my_app_location, c:\%top_level_dir%\%bottom_level_dir% etc.

① Note

- If the environment variable that is used by the configuration parameter is not set to anything, then an exception will be thrown.
- A directory name cannot partially be using an environment variable. For example, c:\my_app_%id%
- Multiple variables can be used in given directory location. For example, c:\%top_level_dir%\%bottom_level_dir%.

① See Also

Oracle Database Net Services Reference chapters covering `sqlnet.ora` parameters, `tnsnames.ora` local naming parameters, and `listener.ora` Oracle Net Listener parameters, for more information about these Oracle client settings.

Oracle Data Provider for .NET Core Configuration

ODP.NET Core developers can assign application settings in .NET Configuration API, `sqlnet.ora` file, and `tnsnames.ora` file.

.NET Configuration API

.NET Core does not support application configuration via .NET configuration files, that is, `web.config`. Instead, it uses .NET Configuration API in lieu of a configuration file. ODP.NET Core supports Configuration API via the static class, `OracleConfiguration`, for application level provider settings. The `OracleDataSourceCollection` class supports adding and deleting net services names, that is, TNS entries. The `OracleOnsServerCollection` class supports adding to and deleting from a list of nodes where the Oracle Notification Service (ONS) daemons are talking to their remote clients.

All configurations settings through `OracleConfiguration` should be done before opening any connection in the application. Once a connection is opened, any updates to configuration properties will result in `InvalidOperationException`; with only exception of trace settings that are still allowed to change during application runtime.

Example 2-6 Code Sample

```
using System;
using Oracle.ManagedDataAccess.Client;

namespace ODP_Core_Config_API
```

```

{
    class odp_core_config
    {
        static void Main(string[] args)
        {
            // This sample demonstrates how to use ODP.NET Core Configuration API

            // Add connect descriptors and net service names entries.
            OracleConfiguration.OracleDataSources.Add("orclpdb",
"(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<hostname or IP>)(PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=<service name>)(SERVER=dedicated)))");
            OracleConfiguration.OracleDataSources.Add("orcl",
"(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<hostname or IP>)(PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=<service name>)(SERVER=dedicated)))");

            // Set default statement cache size to be used by all connections.
            OracleConfiguration.StatementCacheSize = 25;

            // Disable self tuning by default.
            OracleConfiguration.SelfTuning = false;

            // Bind all parameters by name.
            OracleConfiguration.BindByName = true;

            // Set default timeout to 60 seconds.
            OracleConfiguration.CommandTimeout = 60;

            // Set default fetch size as 1 MB.
            OracleConfiguration.FetchSize = 1024 * 1024;

            // Set tracing options
            OracleConfiguration.TraceOption = 1;
            OracleConfiguration.TraceFileLocation = @"D:\traces";
            // Uncomment below to generate trace files
            //OracleConfiguration.TraceLevel = 7;

            // Set network properties
            OracleConfiguration.SendBufferSize = 8192;
            OracleConfiguration.ReceiveBufferSize = 8192;
            OracleConfiguration.DisableOOB = true;

            OracleConnection orclCon = null;

            try
            {
                // Open a connection
                orclCon = new OracleConnection("user id=hr; password=<password>; data
source=orclpdb");
                orclCon.Open();

                // Execute simple select statement that returns first 10 names from
EMPLOYEES table
                OracleCommand orclCmd = orclCon.CreateCommand();
                orclCmd.CommandText = "select first_name from employees where rownum <=
10 ";

                OracleDataReader rdr = orclCmd.ExecuteReader();

                while (rdr.Read())
                    Console.WriteLine("Employee Name: " + rdr.GetString(0));

                Console.ReadLine();
            }
        }
    }
}

```

```

        rdr.Dispose();
        orclCmd.Dispose();
    }
    finally
    {
        // Close the connection
        if (null != orclCon)
            orclCon.Close();
    }
}
}
}
}

```

Oracle Configuration Files

ODP.NET Core supports the `sqlnet.ora` and `tnsnames.ora` parameters below. These settings can be used in conjunction with .NET Configuration API.

- BindByName
- DbNotificationPort
- Disable_Oob – `sqlnet.ora`
- DRCPCConnectionClass
- FetchSize
- HTTPS_PROXY - `tnsnames.ora`
- HTTPS_PROXY_PORT - `tnsnames.ora`
- MaxStatementCacheSize
- NAMES.DIRECTORY_PATH – `sqlnet.ora`
- NODELAY – `sqlnet.ora`
- RETRY_COUNT
- RETRY_DELAY
- RECEIVE_BUF_SIZE – `sqlnet.ora` or `tnsnames.ora`
- SelfTuning
- SEND_BUF_SIZE – `sqlnet.ora` or `tnsnames.ora`
- ServiceRelocationConnectionTimeout
- SQLNET.AUTHENTICATION_SERVICES – `sqlnet.ora`
- SQLNET.CRYPTO_CHECKSUM_CLIENT – `sqlnet.ora`
- SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT – `sqlnet.ora`
- StatementCacheSize
- SSL_SERVER_DN_MATCH – `sqlnet.ora`
- SSL_VERSION – `sqlnet.ora`
- TNS_ADMIN
- TOKEN_AUTH
- TOKEN_LOCATION
- TraceFileLocation

- `TraceLevel`
- `TraceOption`
- `TCP.CONNECT_TIMEOUT` – `sqlnet.ora`
- `SQLNET.ENCRYPTION_CLIENT` – `sqlnet.ora`
- `SQLNET.ENCRYPTION_TYPES_CLIENT` – `sqlnet.ora`
- `WALLET_LOCATION`

ODP.NET Core will look for its connection configuration settings in the following precedence order:

1. `OracleConfiguration.OracleDataSources`
2. Directory set in `OracleConnection.TnsAdmin` property
3. Directory set for the `Tns_Admin` (connection string attribute) or `tns_admin` setting (connection string URL or centralized configuration file connection string).
4. Directory set in `OracleConfiguration.TnsAdmin` property
5. Current working directory
6. `TNS_ADMIN` directory setting of the Windows environment variable or container environment variable
7. `%ORACLE_HOME%\network\admin` directory

ODP.NET Core will look for `ldap.ora` files in the following precedence order:

1. Directory set in `OracleConnection.TnsAdmin`
2. Directory set for the `Tns_Admin` connection string attribute
3. Directory set in `OracleConfiguration.TnsAdmin` property
4. Directory set in `OracleConfiguration.LdapAdmin` property
5. Current working directory
6. `TNS_ADMIN` directory setting in the environment
7. `LDAP_ADMIN` directory setting in the environment

① See Also

- [OracleConfiguration Class](#)
- [OracleDataSourceCollection Class](#)

Configuration Differences among ODP.NET Drivers

[Table 2-9](#) lists other configuration differences among ODP.NET Core, ODP.NET, Managed Driver and ODP.NET, Unmanaged Driver..

Table 2-9 Configuration Differences among ODP.NET Core, ODP.NET, Unmanaged Driver and ODP.NET, Managed Driver

Feature Category	Difference compared to ODP.NET, Unmanaged Driver
Configuration	The older, traditional ODP.NET, Unmanaged Driver configuration file format is different. The new format allows both providers to share the same format. See "Oracle Data Provider for .NET, Managed Driver Configuration."
Configuration	Windows Registry based configuration is not supported
Configuration Parameter	Edition is not supported.
Configuration Parameter	CheckConStatus is not supported.
Configuration Parameter	DllPath is not supported.
Configuration Parameter	StatementCacheWithUdts is not supported.
Configuration Parameter	ThreadPoolMaxSize is not supported.
Configuration Parameter	TraceFileName is not supported.
Configuration Parameter	UdtCacheSize is not supported.
Configuration Parameter	<i>UDT Mapping</i> is not supported.
Configuration Parameter	UseOraMTSManaged is not supported.
Connection String	Context Connection is not supported.
Connection String	LegacyTransactionBindingBehavior setting will be ignored. It will always be set to the default value of 1.
Connection String	Statement Cache Purge is not supported.
Connection String	Tns_Admin is supported in managed ODP.NET and ODP.NET Core, but not in unmanaged ODP.NET.
Connection String	Wallet_Location is supported in managed ODP.NET and ODP.NET Core, but not in unmanaged ODP.NET.
Connectivity	Connection to Oracle Times Ten Database is not supported.
Performance Monitor	NumberOfStaticConnections performance counter is not supported.
Performance Monitor	Performance monitor category name is "ODP.NET, Managed Driver"
Provider Types	Provider Types accept (via constructors) and generate (via ToString() methods) only culture-invariant strings
Tracing	Dynamic tracing is enabled by changing the TraceLevel setting in the app/web/machine.config. NOTE: For ASP.NET applications, doing so will recycle the application domain.

Centralized Configuration Providers for Deployments

Managed ODP.NET and ODP.NET Core can store and retrieve connection string configuration values securely from a centralized provider. Centralizing ODP.NET connection configuration information makes managing user credentials easier in an app deployment. Ensuring connection strings are stored securely is necessary since they can contain sensitive information, such as passwords.

This configuration provider is available for ODP.NET cloud deployments and on-premises deployments and work with any Oracle database version ODP.NET supports. It can be stored in the following locations:

- Microsoft Azure: Store database connection information in Azure App Configuration and Azure Key Vault.
- Oracle Cloud Infrastructure (OCI): Store database connection information in OCI Object Storage and OCI Vault.
- Local file system: Store database connection information on the local file system for web and desktop apps.

Since centralized configuration providers are often used for cloud deployments, local file system access is disabled by default when the connection string is populated with an URL. To enable local access, set `OracleConnection.RemoteConfigurationFiltering` to `false`.

Connection String URLs

The connection string URL is different from the traditional ODP.NET connection string format. To access the Azure, OCI, or local file centralized configuration provider, ODP.NET apps specify a URL in the `OracleConnection.ConnectionString` property in place of the traditional ODP.NET connection string. This URL contains the centralized configuration provider type, that is, Azure, and its configuration storage name, as well as authentication and connection configuration parameter values, if applicable. The traditional connection string values, such as `User Id` and `Min Pool Size` are stored within the centralized configuration provider itself.

Nonetheless, it remains simple moving from a traditional ODP.NET connection string to the centralized configuration provider model. The traditional ODP.NET connection string attributes and values are moved over to Azure App Configuration, OCI Object Storage, or local directory. Then, the `ConnectionString` property is set to the connection string URL instead of a traditional ODP.NET connection string.

Sensitive information, such as passwords, can be stored in secure stores, such as Azure Key Vault and OCI Vault, or be encoded. The centralized configuration providers do not support storing plain text passwords in the connection string.

If using pooling, ODP.NET caches the connection string URL. Changing the configuration values in the centralized storage after the pool populates will not change the connection pool properties until the application is restarted.

If pooling is disabled, the connection string URL is not cached. Changes to the connection configuration is then possible for subsequent connection requests without an application restart.

Connection strings begin with the format `config-<provider type>` where the provider type is `azure`, `ociobject`, or `file`, depending on the deployment location.

Azure Centralized Configuration Provider

The Azure configuration provider can use Azure App Configuration and Azure Key Vault store to store Oracle database connection information. The Azure Centralized Configuration Provider connection string URL syntax is:

```
config-azure://{Config Name}[?  
Key=Prefix&Label=Value&Option1=Value1&Option2=Value2...]
```

- (Required) Config Name: The Azure App Configuration name
- (Optional) Key: a key prefix to identify the connection. Ending the key prefix with a separator, such as `":"`, `"/"` or `"."`, improves readability, but is not a technical requirement. If a key is not specified, a top-level connection is implied.

- (Optional) Label: The Azure App Configuration label name
- (Optional) Options and Values: The Azure authentication and change notification options and information to use when connecting to the Azure App Configuration service. If no authentication information is provided, ODP.NET uses Azure Default Credential to access the Azure App Configuration. More information about Azure change notification is in the [Connection URL Caching](#) documentation.

Azure Authentication Options

ODP.NET uses the Azure SDK for .NET credential classes to provide authorization and authentication to Azure App Configuration and Key Vault services. This dependency can be resolved by having your project incorporate the Oracle.ManagedDataAccess.Azure package available on NuGet Gallery when Azure authentication is required.

Developers set the Azure authentication option to “Authentication” in the ODP.NET URL connection string. The possible values for this option are listed in the following table.

Note

These options and values are case insensitive. The required parameters can be set alternatively as environment variables

Table 2-10 Azure Authentication Methods and their Values

Authentication Method	Authentication Option Values	Required Parameters for This Option Value	Optional Parameters for This Option Value
Default Azure Credential	AZURE_DEFAULT or Empty String		AZURE_TENANT_ID AZURE_MANAGED_IDENTITY_CLIENT_ID
Service Principal with client secret*	AZURE_SERVICE_PRINCIPAL	AZURE_CLIENT_ID AZURE_CLIENT_SECRET AZURE_TENANT_ID	
Service Principal with client certificate*	AZURE_SERVICE_PRINCIPAL	AZURE_CLIENT_ID AZURE_CLIENT_CERTIFICATE_PATH AZURE_TENANT_ID	AZURE_CLIENT_CERTIFICATE_PASSWORD
Managed Identity	AZURE_MANAGED_IDENTITY	AZURE_CLIENT_ID - required only if user assigned	
Interactive Browser OAuth token flow	AZURE_INTERACTIVE	AZURE_CLIENT_ID AZURE_REDIRECT_URL	

* The Azure service principal with client secret overrides Azure service principal with client certificate.

ODP.NET supports Azure service principal with a client secret stored in a user created wallet. ODP.NET first attempts to use the ODP.NET Azure token client secret or connection string URL Azure client secret values. If neither contain a value, ODP.NET will then use a client secret stored in the wallet. The wallet path is specified in the `azure_credentials` property. This property can be set in TNS descriptor or `sqlnet.ora`.

When this property is specified, ODP.NET will attempt to authenticate with Azure by utilizing the secret corresponding to the configured wallet entry of the format `oracle.security.azure.credential.{CLIENT_ID}`, where `{CLIENT_ID}` is the Microsoft Entra ID registered Client identifier.

The `azure_credentials` property is ignored if `client_secret` property is specified using the `AzureTokenAuthentication` class.

The ODP.NET connection string URL provides more connection configuration options, including:

- `AZURE_MAX_RETRIES`
- `AZURE_RETRY_TIMEOUT`
- `AZURE_MANAGED_IDENTITY_CLIENT_ID` - only available when using `DefaultAzureCredential` and for user assigned.
- `AZURE_TENANT_ID`
- `AZURE_AUTHORITY_HOST`

The following code sample provides an ODP.NET connection string URL that specifies an Azure App configuration store name called `MyConfig`. The prefix key for the connection in `MyConfig` is `MyConnection` and the Azure App Configuration label name is `production`. The code sample uses Azure Managed Identity authentication to connect and supplies that id value. The code sample then sets the connection string URL using `OracleConnection` constructor. ODP.NET obtains the connection string from the Azure App Configuration store implicitly when opening the connection.

```
{
  // This ODP.NET URL uses Azure Managed Identity authentication
  string connectionString = "config-azure://MyConfig?key=
    MyConnection/&Label=production&Authentication=Azure_Managed_Identity&
    Azure_Managed_Identity_Client_Id=111abccd8-18cc-4fa0-bf7b-d894e5981c67";
  OracleConnection conn = new OracleConnection();
  conn.ConnectionString = connectionString;
  conn.Open();
}
```

See Also

[DefaultAzureCredential Class](#) for more information about Azure identity and authentication.

OCI Centralized Configuration Provider

The Oracle Cloud Infrastructure configuration provider uses OCI Object Storage and OCI Vault to store database connection information in a JSON configuration file. The connection string URL syntax is:

```
config-ociobject://{Object-Url}[?key=name&option1=value1&option2=value2...]
```

- (Required) `Object-Url`: The object URL path in OCI Object Storage. This value can be obtained in the OCI Web Console in the Object Details of the Object Storage / Buckets / Object menu. Two examples of Object-Urls are:

```
https://namespace.objectstorage.us-ashburn-1.oci.customer-oci.com/n/
namespace/b/bucket1/o/odp.json
```

and,

```
https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace/b/bucket1/o/
odp.json.
```

- (Optional) Key: The connection key name identity. If a key is not specified, a top-level connection is implied.
- (Optional) Options: The Oracle Cloud authentication and change notification options and information to use when connecting to OCI Object Storage. If no authentication is provided, then the provider uses OCI API Key-Based Authentication to access OCI Object Storage. More information about Oracle Cloud change notification is in the Connection URL Caching documentation.

OCI Authentication Options

ODP.NET OCI authentication has a dependency on OCI SDK for authorization and authentication to the OCI Object Storage and Vault services. This dependency can be resolved by having your project incorporate the `Oracle.ManagedDataAccess.Oci` package available on NuGet Gallery when OCI authentication is required.

The authentication parameters are provided via the connection string URL.

Developers set the OCI authentication option to `Authentication` in the ODP.NET URL connection string. The possible values for this option are listed in the following table.

Note

These options and values are case insensitive. The required parameters can be set alternatively as environment variables

Table 2-11 OCI Authentication Methods and their Values

Authentication Method	Authentication Option Values	Optional Configuration	Optional Parameters
API Key-Based Authentication	OCI_DEFAULT or Empty String	(~/oci/config), (~/oraclebmc/config), or environment variable OCI_CONFIG_FILE	OCI_PROFILE OCI_TENANCY OCI_USER OCI_FINGERPRINT OCI_KEY_FILE OCI_PASS_PHRASE
Instance Principal Authentication	OCI_INSTANCE_PRINCIPAL		
Resource Principal Authentication	OCI_RESOURCE_PRINCIPAL		

The following code sample provides an ODP.NET connection string URL that specifies an OCI Object Store object URL. The connection key name identity is `MyConnection`. The code sample uses OCI's API key-based authentication, `OCI_DEFAULT`, and the optional parameter, `OCI_PROFILE`, to specify the connection profile configuration. The code sample then sets the connection string URL using `OracleConnection.ConnectionString` property. ODP.NET obtains

the connection string from the JSON configuration file stored in OCI Object Storage Service when opening the connection.

```
{
    //This URL will use default OCI authentication, which requires no authentication
parameters
    string connectionString = "config-ociobject://https://objectstorage.us-ashburn-1.
    oraclecloud.com/n/namespace/b/bucket1/o/odp.json?
key=MyConnection&authentication
    =OCI_DEFAULT&OCI_PROFILE=MyProfile";
    OracleConnection conn = new OracleConnection(connectionString);
    conn.Open();
}
```

See Also

[OCI SDK Authentication Methods](#) for more information.

Local File Centralized Configuration Provider

The file config provider stores Oracle database connection information in a JSON configuration file on the local file system. The ODP.NET file connection string URL syntax is:

config-file://{File Path}[?Key=name&tns_admin=filepath]

- (Required) File Path: JSON config file path and name, which can be an absolute or relative path. For relative paths, ODP.NET will use the TNS_ADMIN value as the base path.
- (Optional) Key: The connection key name identity. If a key is not specified, a top-level connection is implied.
- (optional) tns_admin: the filename base path or prefix to locate the JSON config file.

The following code sample provides an ODP.NET connection string URL that specifies a local file, odpconfig.json, and directory location. The connection key name identity is MyConnection. This code sample then sets the connection string URL using OracleConnection constructor. ODP.NET obtains the connection string from the local JSON config file when opening the connection.

```
{
    //This URL will retrieve the c:\configs\odpconfig.json from the file system
    //The connection key is "MyConnection"
    string connectionString = "config-file://c:/configs/odpconfig.json?
key=MyConnection";

    OracleConnection conn = new OracleConnection(connectionString);
    conn.Open();
}
```

The file config provider relies on Oracle.ManagedDataAccess.ConfigFile package available on NuGet Gallery.

JSON Configuration File Search Order

ODP.NET can use the absolute directory and file name value, that is, c:/configs/myconfig.json, set in the local file centralized configuration provider connection string URL to

locate its JSON configuration file. It can also use a relative directory with a file name. In this case, it will append the relative directory and file to the settings in the following order to locate the JSON file.

1. Current working directory
2. Directory set in `OracleConnection.TnsAdmin` property
3. Directory set for the `Tns_Admin` connection string attribute
4. Directory set in the `tns_admin` attribute of the configuration provider URL
5. Directory set in `OracleConfiguration.TnsAdmin` property
6. `TNS_ADMIN` directory setting in .NET configuration file -- managed ODP.NET only
7. `TNS_ADMIN` directory setting of the Windows environment variable or container environment variable
8. Directory set in the Windows Registry key, `HKLM\SOFTWARE\Oracle\ODP.NET.Managed\<version>\TNS_ADMIN` -- managed ODP.NET only

Note

32-bit apps use the Registry key
`HKLM\SOFTWARE\Wow6432Node\Oracle\ODP.NET.Managed\<version>\TNS_ADMIN`

9. `%ORACLE_HOME%\network\admin` directory

Vault Providers

Database passwords, proxy passwords, wallets, JSON configuration are sensitive database information that can be stored in Azure Key Vault or OCI Vault services as secret. These vault providers are included with the `Oracle.ManagedDataAccess.Azure` and `Oracle.ManagedDataAccess.Oci` NuGet packages, respectively.

ODP.NET provides a secured way to access this sensitive information from vaults using a predefined URL, secret-URL. The secret-URL must be prefixed with `vault-azure` or `vault-oci` for ODP.NET to access the user provided information in either Azure Key Vault or OCI Vault services, respectively. The user-provided URL must conform to the ODP.NET defined syntax below.

The ODP.NET vault providers support single sign-on (.sso) and PKCS #12 (.p12) wallets in cloud vaults.

Azure Key Vault

All ODP.NET secrets can be stored as Azure Vault secrets. Use the following URL syntax to access Azure Key Vault secrets:

```
config-azurevault://{Secret Identifier}[?key=name&Option1=Value1&Option2=Value2...]
```

Applications need to provide the following information in the URL:

- (Required) Secret Identifier: The Azure secret identifier obtained from the Azure portal
- (Optional) Key: The connection identifier. If a key is not specified, then it is a top level connection. This option is only used when a JSON configuration file is in Azure Key Vault.

- (Optional) Authentication Options: The Azure Key Vault authentication connection information. If authentication information is not provided, ODP.NET uses Azure Default Credential to access secrets in Azure Key Vault service. The authentication options are the same as the [Azure Authentication Options](#).

OCI Vault

All ODP.NET secrets can be stored as OCI Vault secrets. Use the following URL syntax to access OCI Vault secrets:

```
config-ocivault://{Secret OCID}[?
VersionNumber=Value1&Stage=Value2&key=name&Option3=Value3&Option4=Value4...]
```

Applications need to provide the following information in the URL:

- (Required) Secret OCID: The OCI secret identifier (OCID) obtained from the OCI portal
- (Optional) VersionNumber: The OCI secret version. If the version is not specified, the latest secret version is retrieved by default.
- (Optional) Stage: The secret's rotation stage. If stage is not specified, the secret's current stage is retrieved by default.
- (Optional) Key: The connection identifier. If a key is not specified, then it is a top level connection. The option is only used when a JSON configuration file is in the OCI Vault.
- (Optional) Options: The OCI Vault authentication connection information. If authentication information is not provided, ODP.NET uses the API Key-Based authentication to access secrets in OCI Vault service. The authentication options are the same as the [OCI Authentication Options](#).

Vault Connection String URLs

Similarly for centralized configuration providers, vault URLs can be passed as a connection string to the `OracleConnection` class through a constructor connection string property or an `OracleConnection.ConnectionString` property.

Here is an example for Azure Vault:

```
new OracleConnection("config-azurevault://https://vault1.vault.azure.net/secrets/
ODPConnStr/
abc69352e1d040e59wd7b83ffcdaf123");
```

The Vault can contain a standard connection string or a JSON config string.

Standard connection string example:

```
user id = scott; password = tiger; data source = inst1; pooling = false
```

JSON config example:

```
{
  "Connection1": {
    "ODP": {
      "User Id": "scott",
      "Password": "tiger",
      "Data Source": "inst1",
      "Pooling": "false"
    }
  }
}
```

Note

The connection/JSON config strings should be uploaded as clear text.

Vault Passwords

Instead of storing the entire connection string, developers have the option to store only the password or proxy password in a cloud vault. To do this, the application provides the vault URL value in the password or proxy password connection string attribute value.

For example, the ODP.NET connection string for Azure Vault could be:

```
"User ID = HR; Data Source = mydb;  
password=config-azurevault://https://vault1.vault.azure.net/secrets/ODPConnStr/  
abc69352e1d040e59wd7b83ffcdaf123"
```

Vault Wallets and Tokens

An application can store wallets and tokens in cloud vault services securely by providing the vault URL in the ODP.NET wallet and token location properties. The application must store the wallet or token file data in the cloud vault as a base64 string.

Note

For OCI vault, although wallet and token have to be converted to base64 string, they must be uploaded to OCI vault as "plain-text" type secret.

ODP.NET connection strings can use the `Wallet_Location` and `Token_Location` properties, respectively, to store wallets and tokens. For example, the ODP.NET connection string for accessing a wallet stored in Azure Vault could be:

```
"User ID = HR; password = <PASSWORD>; Data source = mydb;  
Wallet_Location=config-azurevault://https://vault1.vault.azure.net/secrets/HRWallet/  
abc69352e1d040e59wd7b83ffcdaf123"
```

The `OracleConnection.WalletLocation` and `TokenLocation` properties can be set to a vault URL or file path location. These `OracleConnection` properties have precedence over the wallet and token location attributes in the connection string. Here's sample code to use these properties with Azure Vault:

```
OracleConnection conn = new OracleConnection();  
conn.WalletLocation = "config-azurevault://https://vault1.vault.azure.net/secrets/  
HRWallet/abc693...";  
conn.TokenLocation = "config-azurevault://https://vault1.vault.azure.net/secrets/HRToken/  
abc3ffc...";
```

Note

Secure External Password Store is not supported for cloud wallet locations.

ODP.NET supports Microsoft identity platform tokens v1.0 for Microsoft Entra ID-only applications and v2.0 for applications that support consumer accounts. ODP.NET 23.5 and higher check the roles (roles) and user principal name (UPN) claims in v2.0 tokens to verify at least one is present. If both claims are missing, then the provider will report an error.

Provider Configuration

The local file and OCI centralized configuration providers both use a JSON format to specify connection information. They both use the same JSON configuration.

Provider Configuration Keys

ODP.NET uses the values in the provider configuration keys to construct a connection string to an Oracle database. Configuration keys generally map to an existing ODP.NET connection string and Oracle connection properties. Some keys are generic to all Oracle data access drivers. Those properties are:

- user: database user/schema name
- password: database password
- connect_descriptor: ODP.NET data source
- config_time_to_live: cached connection information retention time in seconds
- config_time_to_live_grace_period: After config_time_to_live expires, new connections can continue to use the cached credentials for the set grace period.

In Azure App Configuration, the ODP.NET specific configuration keys are prefixed with ODP/ to indicate intent to be used for ODP.NET only. For example, MyConnection/ODP/Pooling key specifies the ODP.NET's Pooling connection string attribute.

For OCI and local file providers, the ODP.NET specific configuration keys are specially identified in the JSON configuration. These keys use the "ODP" key. For example, "ODP": {"Max Pool Size": "500", "Pooling": true } specifies the Max Pool Size and Pooling connection string attributes for MyConnection property in the following example.

```
{
  "MyConnection": {
    "ODP": {"Max Pool Size": "500", "Pooling": true },
    "password":
    {
      "type": "ocivault",
      "value": "ocid1.vaultsecret.oc1.phx.example",
      "authentication":
      {
        "method": "OCI_DEFAULT",
        "OCI_PROFILE": "<PROFILE>"
      }
    }
  },
  "user": "HR",
  "connect_descriptor": "(description= .....)"
}
```

Generally, ODP.NET configuration keys are case insensitive. However, the ODP/ section in Azure App configuration and the ODP section in OCI and local configuration, including the keys and their values, are case-sensitive.

Connection Attributes

The `ConnectionString` key is an ODP.NET specific key in the configuration provider. It allows defining complete or partial ODP.NET connection strings. ODP.NET apps can continue to use the `OracleConnection.ConnectionString` property and are not required to use the configuration provider's `ConnectionString` key.

All the connection string attributes defined in `OracleConnection.ConnectionString` are supported as is used in the traditional connection string except the following attribute:

- `Tns_Admin`: Not supported.

ODP.NET also supports the following `OracleConnection` properties in the provider configuration:

- `AutoProxy`
- `BindByName`
- `CommandTimeout`
- `ChunkMigrationConnectionTimeout`
- `ConnectionIdPrefix`
- `DatabaseEditionName`
- `DrcpConnectionClass`
- `FetchSize`
- `HttpsProxy`
- `HttpsProxyPort`
- `MaxStatementCacheSize`
- `OciCompartment`
- `OciDatabase`
- `OciIamUrl`
- `OciTenancy`
- `PasswordAuthentication`
- `Pipelining`
- `PoolName`
- `ServiceRelocationConnectionTimeout`
- `SSLVersion`
- `SSLServerCertDN`
- `SSLServerDNMatch`
- `SuppressGetDecimalInvalidCastException`
- `SqlNetEncryptionClient`
- `SqlNetEncryptionTypesClient`
- `SqlNetCryptoChecksumClient`
- `SqlNetCryptoChecksumTypesClient`

- TokenAuthentication
- UseClientInitiatedCQN
- UseSNI
- Wallet_Location: Wallet that is stored in local system file is not supported. Wallet that is stored as Base64 string in Azure App Configuration, Azure Vault, or OCI Vault is supported for .p12 and .sso files. For JSON configuration, Wallet_Location must be specified using Password JSON object.
- Wallet_Password: For encrypted PKCS #12 wallets.
- Token_Location: Token that is stored in local system file is not supported. Token that is stored as Base64 string in Azure App Configuration, Azure Vault, or OCI vault is supported. For JSON configuration, Token_Location must be specified using Password JSON object.

If a non-supported attribute is specified in the provider configuration under the ODP JSON property or ODP/ Azure App Configuration section, then ODP.NET throws an exception.

The following example contains three connections: sales_app, hr_app, and sh_app using various configuration key types.

```
{
  "sales_app":
  {
    "connect_descriptor": "(description=(address=(protocol=tcps)(port=1521)
(host=<HOSTNAME>
(connect_data=(service_name=<SERVICE NAME>))(security=(ssl_server_dn_match=yes)))",
    "user": "sales",
    "password": {
      "type": "ocivault",
      "value": "<VAULT LOCATION>",
      "authentication": {
        "method": "OCI_INSTANCE_PRINCIPAL"
      }
    }
  },
  "ODP": {
    "Connection Timeout": 30,
    "pooling": "true",
    "max pool size": 100
  }
},
  "hr_app":
  {
    "password": {
      "type": "azurevault",
      "value": "<VAULT LOCATION>",
      "authentication": {
        "method": "AZURE_DEFAULT"
      }
    }
  },
  "ODP": {
    "user": "hr",
    "data source": "oracle",
    "max pool size": 100
  }
},
  "sh_app":
  {
    "ODP": {
      "ConnectionString": "user id = sh; data source = oracle; max pool size = 100",
```

```

        "BindByName": "false"
    }
    "password": {
        "type": "ocivault",
        "value": "<SECRET OCID>",
        "authentication": {
            "method": "OCI_INSTANCE_PRINCIPAL",
            "OCI_PROFILE": "DEFAULT"
        }
    }
}
}
}

```

Within the configuration provider, `ConnectionString` key has precedence over other keys, such as `connect_descriptor`, with the same prefix key. Thus, if the `User Id` attribute is not included in the `ConnectionString`, then ODP.NET uses the user key that has the same prefix key. If the `Data Source` attribute is not included in the `ConnectionString`, then it will use the value of the `connect_descriptor` key that has the same prefix key. Similar behavior applies for all the `ConnectionString` attributes.

The precedence order for constructing the ODP.NET connection string from higher precedence to lower is as follows:

1. ODP.NET `ConnectionString` centralized configuration provider key
2. ODP.NET `OracleConnection` `ConnectionString` attributes, `OracleConnection` properties, provider configuration file, and ODP.NET connection parameters, such as `ODP/Pooling`, `ODP/Connection Timeout`, and `ODP/OciCompartment`. All of these have equal precedence.
3. Oracle common driver attributes in provider configuration file, such as user, password, and `connect_descriptor`.

Password Connection Attributes

For the OCI and local file providers, passwords, wallet locations, and token locations must be specified as password JSON objects. Password JSON objects are supported with Azure Vault as well, but not Azure App Configuration.

The Oracle user password has the following format and properties in the configuration providers:

Property	Required or Optional	Possible Values and Notes
type	Required	- Base64 - azurevault - ocivault
value	Required	- Base64 encoded password (base64) - Azure Key Vault URI (azurevault) - OCID of the secret (ocivault)
authentication	Optional	- Authentication type - Optional parameters This property has the same possible values and defaults as described in the Azure and OCI Centralized Configuration Provider sections.

Property	Required or Optional	Possible Values and Notes
versionNumber	Optional	Applicable to OCI Vault only
stage	Optional	Applicable to OCI Vault only

Sample OCI password configuration:

```
"password": {
  "type": "ocivault",
  "value": "<SECRET OCID>",
  "authentication": {
    "method": "OCI_INSTANCE_PRINCIPAL",
    "OCI_PROFILE": "DEFAULT"
  }
}
```

Sample Azure password configuration:

```
"password": {
  "type": "azurevault",
  "value": "<AZURE KEY VAULT URI>",
  "authentication": {
    "method": "AZURE_SERVICE_PRINCIPAL",
    "AZURE_CLIENT_ID": "<AZURE CLIENT ID>",
    "AZURE_CLIENT_CERTIFICATE_PATH": "<AZURE CLIENT CERTIFICATE PATH>",
    "AZURE_TENANT_ID": "<AZURE TENANT ID>"
  }
}
```

Sample Base64 password configuration

```
{
  "type": "base64",
  "value": "<BASE64 ENCODED PASSWORD>"
}
```

ODP.NET ignores plain text passwords in the OCI Object Store or local file centralized configuration provider file for security reasons. The Azure centralized configuration provider accepts plain text passwords in password keys and the connection string since Azure provides secure means to access the app configuration.

While Base64 format is allowed in the password JSON object, it is not considered secure and should only be used in development environments.

If Persist Security Info is false (default) in the ODP.NET connection string, then the Password value is not returned with the connection string value to protect the sensitive password information. In the Azure case, the following sensitive authentication information is also removed from the connection string:

- AZURE_CLIENT_SECRET
- AZURE_CLIENT_CERTIFICATE
- AZURE_CLIENT_CERTIFICATE_PASSWORD

In the OCI case, the following sensitive authentication information is also removed from the connection string:

- OCI_FINGERPRINT
- OCI_PRIVATE_KEY
- OCI_PASS_PHRASE

Automatic Password Refresh

Administrators often set up password expiration policies so that the passwords become invalid at a regular cadence and replaced with new ones. For existing connection pools, this can result in ORA-01017 errors (invalid username/password; logon denied) and having to rebuild the connection pool.

ODP.NET supports the ability to automatically refresh the existing connection pool with the new password. This feature eliminates end users experiencing connection errors and forcing apps to rebuild their pool after each password refresh.

Using this feature requires:

- ODP.NET connection with pooling enabled
- An URL connection string stored in a centralized configuration provider
- One of the following locations to specify the password:
 - “ODP/ConnectionString” key
 - “ODP/Password” key
 - “ODP/ProxyPassword” key
 - “Password” key

Note

ODP.NET does not currently support password refresh using cloud key vaults.

When ODP.NET encounters an invalid password, it retrieves the latest configuration provider values and checks for the following conditions:

- The new password(s) must use the same key(s) as the expired password(s).
 - If the original configuration used the “ODP/Password” key, but the refreshed configuration uses the “Password” key, the original connection error will be thrown.
 - If the original configuration used the “ODP/Password” and “ODP/ProxyPassword” keys, but the refreshed configuration only uses the “ODP/Password” key, the original connection error will be thrown.
- The password value must be different between the original and updated passwords.
 - If both the password and proxy password are used, then at least one of them must be different.
- All non-password related keys must be identical between the original and updated values.

If any part of this process does not meet the stated conditions, then ODP.NET returns the original connection error.

For end users and administrators, this automated password refresh experience provides a seamless and error free experience.

Connection URL Caching

Since connection information retrieval takes longer retrieving from a cloud resource than local disk, ODP.NET can cache connection information. Cache updates can occur using a polling or change notification model.

Connection URL Polling Updates

The polling model uses the `config_time_to_live` key. This property sets the duration in seconds for ODP.NET to keep the connection information cached.

By default, `config_time_to_live` is 86,400 seconds (24 hours). The maximum value is `Int64.MaxValue`. If it is set to a non-positive value, ODP.NET will never expire the cache.

When the cache expires, ODP.NET retrieves the connection information from the configuration provider again. Once retrieved, ODP.NET refreshes the cache and uses the new configuration for the duration of the `config_time_to_live` setting.

Note

If pooling is enabled, then only the password and proxy password can be updated. If pooling is disabled, then all the connection information will be refreshed.

When `config_time_to_live` expires, the configuration is considered "softly expired". ODP.NET attempts to refresh the cache with every connection request. If a new configuration cannot be retrieved, then ODP.NET continues to open new connections with the "softly expired" configuration for up to the time set in `config_time_to_live_grace_period`.

By default, `config_time_to_live_grace_period` is 1800 seconds (30 minutes). It is an `Int64`. Any negative value will be treated as a grace period of zero seconds.

If new connection information can not be retrieved during `config_time_to_live_grace_period`, then the configuration fully expires. No new connections can be made with these old credentials. Existing "softly expired" connections will continue to work until closed or disposed.

Connection URL Change Notification Updates

The change notification model provides real time connection information updates. When data on the cloud is modified, ODP.NET receives the notification almost immediately instead of waiting for `config_time_to_live` to expire. The relevant cache entries are evicted, and the latest values are retrieved the next time they are needed.

Connection URL change notifications are available for Azure App Configuration and OCI Object Storage only at this time.

Similarly to `config_time_to_live`, if connection pooling is disabled, then change notification can monitor and update all connection configuration provider settings. If pooling is enabled, only the password and proxy password are monitored and updated.

Developers register their apps for change notification using the options in the connection string URL.

To unregister from change notification, developers call the `OracleConnection.UnregisterCloudConfigNotification(OracleConnection con)` method. If

pooling is enabled, the method unregisters all connections that reside in the same pool as the connection passed into the method.

Change notification updates do not reset the `config_time_to_live` timer.

Azure App Configuration Setup for Change Notification

ODP.NET Azure-based change notifications require Azure Event Grid and Azure Service Bus. Azure Event Grid watches the Azure App Configuration for changes. When a change is detected, it sends a notification message to Azure Service Bus. Service Bus forwards the message to the ODP.NET application, prompting the application to refresh the values associated with the update notification upon their next use.

Azure Service Bus

Azure Service Bus is a fully managed enterprise message broker with message queues and publish-subscribe topics. ODP.NET change notification can use the Service Bus message entities, Queues and Topics. Queues follow a one-to-one model, where one sender delivers messages to one receiver. Topics follow a one-to-many model, where one sender delivers messages to multiple receivers.

Topics are used when multiple ODP.NET applications register for change notification on the same Azure App Configuration. Otherwise, Queues are sufficient. Azure Service Bus pricing varies depending on the usage and feature set the change notification app required.

Azure Service Bus Setup

1. On the Azure portal, navigate to the Service Bus page.
2. Start a new Service Bus resource setup by selecting a pricing tier appropriate for the intended use and budget.
3. Once the Service Bus resource is created, select it, and go to the Overview tab. Remember the "Host name"
 - It will look like: `<Service Bus Name>.servicebus.windows.net`

Next, you will need to set up either Service Bus queue or topic, not both.

Azure Service Bus Queue Setup

1. On the Azure portal Service Bus page, navigate to the Queues and create a new Queue.
 - The default options can be used or be modified. It will not affect compatibility with ODP.NET change notifications.
2. Name the Queue and remember the Queue name.

Azure Service Bus Topic Setup

1. On the Azure portal Service Bus page, navigate to the Topics and create a new Topic.
 - The default options can be used or be modified. It will not affect compatibility with ODP.NET change notifications.
2. Name the Topic and remember the Topic name.
3. Navigate to the created topic.
4. Create a new subscription.
 - The default options can be used with ODP.NET change notifications.
5. Name the Subscription and remember the Subscription name.

- Repeat Steps 4 and 5 if there are multiple ODP.NET Applications that will receive change notifications from the same Azure App Configuration. Each application should use a separate subscription.

Azure Event Grid Setup

To connect the App Configuration to Service Bus, an Event Grid resource is required.

- On the Azure portal, search or navigate to the Event Grid page.
- Go to the "System topics" tab and create a new Event Grid System Topic.
- For "Topic Types", search for and select Microsoft Azure App Configuration
- Have the System Topic point to the App Configuration to be watched.
- Navigate to your new System Topic and create a new subscription.
- For "Event Schema", select "Event Grid Schema" from the drop-down menu.
- For "Filter to Event Types", select both "Key-Value Modified" and "Key-Value Deleted".
- For "Endpoint Type", select "Service Bus Topic" or "Service Bus Queue" depending on whether you are using a topic or queue.
- For the "Endpoint", select the topic or queue created earlier.
 - For the other options, you may use the default values.

ODP.NET Setup for Azure Change Notifications

To enroll ODP.NET in change notifications, the following options in the URL connection string are required for Service Bus queues:

Service Bus Option	Description
SERVICE_BUS_HOST	Service Bus host URL
SERVICE_BUS_QUEUE	Service Bus queue name

All options are required values for queues and all are string values.

URL Connection String format for Azure queues

```
string connStr = "config-azure://myAppConfig?key=test/&SERVICE_BUS_HOST=<Service Bus Host>
&SERVICE_BUS_QUEUE=<Queue Name>";
```

For Service Bus topics, the following options in the ODP.NET URL connection string are required:

Service Bus Option	Description
SERVICE_BUS_HOST	Service Bus host URL
SERVICE_BUS_TOPIC	Service Bus topic name
SERVICE_BUS_SUBSCRIPTION	Subscription name within the Service Bus topic

All options are required values for topics and all are string values.

URL Connection String format for Azure topics

```
string connStr = "config-azure://myAppConfig?key=test/&SERVICE_BUS_HOST=
<Service Bus Host>&SERVICE_BUS_TOPIC=<Topic Name>&SERVICE_BUS_SUBSCRIPTION=
<Subscription Name>";
```

You have now completed change notification setup for Azure App Configuration.

OCI Object Storage Event Notification for Change Notification

ODP.NET OCI Object Storage-based change notifications require Oracle Cloud Infrastructure Events Service and Notifications service to establish a communication channel that publishes messages using topics and subscriptions. When a change is detected, it sends a notification message to the HTTPS endpoint. The OCI event listener of the ODP.NET application receives the message, prompting the application to refresh the values associated with the update notification upon their next use.

To use OCI Object Storage event notifications, you must configure the OCI Object Storage, OCI Notifications service, event rules, an event listener, and ODP.NET to use OCI event notifications. Let's walk through each of these setup topics.

OCI Object Storage Setup

The first step is to setup the OCI JSON configuration file in OCI Object Storage and then connect to the Oracle database using the file.

1. Create a bucket in Object Storage and enable "Emit Object Events" for that bucket. Existing buckets can enable "Emit Object Events" in the "Bucket Details" > "Bucket Information" section.
2. Upload your OCI JSON configuration file to this bucket.

OCI Event Listener Setup

ODP.NET's OCI Event Listener uses HTTPS protocol to receive notification messages, and it has two requirements. First, OCI Notifications service requires a publicly accessible URL with a certificate authority issued certificate. It also must accept POST calls. The OCI API Gateway can be set up to fulfill this requirement through setting up a route. A gateway route is the mapping between a path, one or more methods, and a back-end service.

Second, the host running ODP.NET application must be able to receive the HTTPS message. This usually requires the host to have a port open to the traffic of the notifications service. If your application is running in an OCI Compute Instance and are using OCI API Gateway, you can modify the virtual cloud network's ingress rules security list to permit an open port to OCI API Gateway for the OCI Notifications service. Remember the network path and port you are forwarding into. You will use it in your ODP.NET connection string URL.

OCI Notifications Service

When the JSON configuration file in the OCI Object Storage bucket is updated, OCI Notifications service can emit an event. The event can publish a notification and use the notification to update to the latest configuration version.

OCI Notifications service supports alarms, event rules, and connectors to emit human-readable messages through supported endpoints, including email, SMS, and Slack messages with an HTTPS payload. Additionally, you can automate tasks through custom HTTPS endpoints and OCI Functions, as well as directly publish messages.

The notifications service can be enabled to set up communication channels for publishing messages using topics and subscriptions. A topic is a communication channel for sending messages to subscriptions. When a message is published to a topic, the notifications service sends the message to all the topic's subscriptions. A subscription is a topic endpoint.

Setup OCI Notification Topic and Subscription

A notification topic allows creating a set of notification subscriptions that can be tied together by an event. Here's how to setup a topic and subscription.

1. Navigate to your OCI tenancy compartment's Notifications service management web page. The page title should be "Topics in <Compartment Name> Compartment".
2. Click the **Create Topic** button.
3. Give the topic an identifiable name. It can be helpful to include bucket/configuration file name and subscription protocols in the name.
4. Click on the newly created topic name in the list. Now, let's create the subscription.
5. Click the **Create Subscription** button.
6. Select **HTTPS (Custom URL)** as the communication protocol and add the URL for HTTPS.
7. Confirm your subscription. For HTTPS, you will receive a payload like the following:

```
{"ConfirmationURL":"https://cell1.notification.us-phoenix-1.oci.oraclecloud.com/20181201/subscriptions/ocid1.onssubscription.oc1.phx.../confirmation?token=MDA...&protocol=CUSTOM_HTTPS"}
```

After confirming your subscription, its status will change from "Pending" to "Active". You can test your subscription by publishing a test message.

Setup OCI Notification Event Rules

Rules trigger actions for specified event conditions. Let's create a rule to trigger a notification when the config file is modified.

1. Navigate to your OCI tenancy compartment's event service rules web page. The page title should be "Rules in <Compartment Name> Compartment".
2. Click the **Create Rule** button.
3. Give the rule an identifiable display name.
4. Under the rule conditions section, set the following values:
 - Condition: Event Type
 - Service Name: Object Storage
 - Event Type: Object – Update

These settings indicate whenever there is an Object Storage object update, an action is triggered. Since update can occur for any object in any bucket, let's add more conditions to narrow it down.

5. Click the **Another Condition** button to add more conditions.
 - a. Create a condition with the following attributes:
 - Condition: Attribute
 - Attribute Name: <Bucket Name or ID>
 - Attribute Value: <Configuration File Bucket Name/OCID>
 - b. Create a second condition with the following attributes:
 - Condition: Attribute
 - Attribute Name: <Resource Name or ID>

- Attribute Value: *<Configuration File Name/OCID>*
6. In the Actions section, set the following values:
 - Action Type: Notifications
 - Notifications Compartment: *<Compartment Name of the Notification Topic>*
 - Topic: *<Topic Name>*
 7. Click the **Create Rule** button.
 8. Click the **Validate Rule** button to test whether the rule works. A notification message should be sent.
 - a. In the Example Events section, use a code editor to change the following settings to the matching values you just set for the event rules.
 - *<Compartment Name of the Notification Topic>*
 - *<Resource Name>*
 - *<Bucket Name>*
 - b. Click the **Check If Example Event Matches Rule** button.

That completes the OCI setup. Let's complete the ODP.NET setup for OCI notifications.

ODP.NET Setup for OCI Notifications

For OCI Object Storage notifications, the ODP.NET connection string URL requires the OCI profile name in the JSON configuration file and the OCI event listener path and port. The connection string URL syntax is:

```
@config-ociobject:{object-url}[?key=name&option1=value1&option2=value2...]
```

To enable OCI change notification, the following options in the URL connection string are required:

OCI Event Listener Option	Description
Event_Listener_Path	network path listener listens on for notifications
Event_Listener_Port	network port listener listens on for notifications

In our setup, the connection string URL will also include the `OCI_PROFILE` name that you created earlier for the OCI configuration file. Your ODP.NET connection string will look similar to the following.

Sample Code: ODP.NET OCI change notification connection string URL

```
OracleConnection conn = new OracleConnection("@config-ociobject:https://objectstorage.us-ashburn-1.oraclecloud.com/.../Connection.json?KEY=myconn&authentication=OCI_DEFAULT&EVENT_LISTENER_PATH=<HOSTNAME/DIRECTORY>&EVENT_LISTENER_PORT=80")
```

You have now completed change notification setup for OCI Object Storage.

Connection Configuration Restriction

ODP.NET has many ways to configure and secure connections to the Oracle database. This flexibility provides numerous options for how apps prefer to connect. In some situations, apps

may want to restrict other connection methods that it does not use, have not been tested, and to limit possible unauthorized entry when they get deployed.

ODP.NET connection configuration restriction allows developers to globally set restrictions for how an app can connect. For example, a SaaS app can allow using MY_WALLET_LOCATION, WALLET_LOCATION, and TOKEN_LOCATION to set a wallet and token directory location. ODP.NET apps would then be allowed to connect with any of these settings in the Data Source connection string value, which is generally either a connect descriptor, Easy Connect string, or TNS alias.

If any other data source parameter is not on the allowed list, the connection request is rejected, and an error is thrown. The error is an "ORA-50122: The following configuration parameters are not allowed to be set:" followed by a parameter name list used but not allowed.

This feature is available in managed ODP.NET and ODP.NET Core only.

Connection configuration uses an allow list, which details to ODP.NET which connection options are allowed. The app can supply the list as a string or file so that code changes are not necessary.

Connection Configuration Restriction Methods

To use connection configuration restriction, a developer can call the following OracleConfiguration methods to set or unset the restrictions:

- `EnableConnectionStringAllowedProperties(FileInfo, bool)`
This method accepts a file path to an allowed parameter list in a JSON file. The file should be named `ODPConnectionStringAllowedProperties.json`. Developers may configure whether to enforce allowed parameters on the TNS alias by setting the Boolean argument.
- `EnableConnectionStringAllowedProperties(string, bool)`
This method accepts an allowed parameter list in a JSON string format. Developers may configure whether to enforce allowed parameters on the TNS alias by setting the Boolean argument. If both overloaded methods are set, this JSON string method will override using the JSON file.
- `DisableConnectionStringAllowedProperties`
This method disables using any prior list of allowed parameters. Connections are no longer subject to configuration restrictions.

Data Source Allowed Parameters

The connection configuration JSON format to allow parameters is:

```
{
  "DataSource" : {
    "<property_name1>" : true/false,
    "<property_name2>" : true/false,
    "<property_name3>" : true/false
  }
}
```

This sample JSON file lists all properties that can be disallowed. Each of the properties set to true is allowed. Properties set to false are restricted. Connections using those restricted properties will be rejected.

All properties not set are false by default.

If there are duplicate property names, then the last property name is used.

The JSON strings are case insensitive.

ODP.NET Connection Configuration JSON Sample

```
{
  "DataSource" : {
    "CLIENT_ID" : true,
    "COLOCATION_TAG" : true,
    "CONNECT_TIMEOUT" : true,
    "CONNECTION_ID_PREFIX" : true,
    "ENABLE" : true,
    "EXPIRE_TIME" : true,
    "FAILOVER" : true,
    "HOST" : true,
    "HTTPS_PROXY" : true,
    "HTTPS_PROXY_PORT" : true,
    "IGNORE_ANO_ENCRYPTION_FOR_TCPS" : true,
    "INSTANCE_NAME" : true,
    "LOAD_BALANCE" : true,
    "PASSWORD_AUTH " : true,
    "POOL_BOUNDARY" : true,
    "POOL_CONNECTION_CLASS" : true,
    "POOL_NAME" : true,
    "POOL_PURITY" : true,
    "PORT" : true,
    "PROTOCOL" : true,
    "RECV_BUF_SIZE" : true,
    "RECV_TIMEOUT" : true,
    "RETRY_COUNT" : true,
    "RETRY_DELAY" : true,
    "SDU" : true,
    "SEND_BUF_SIZE" : true,
    "SERVER" : true,
    "SERVICE_NAME" : true,
    "SHARDING_KEY" : true,
    "SHARDING_KEY_ID" : true,
    "SOURCE_ROUTE" : true,
    "SSL_CERTIFICATE_THUMBPRINT" : true,
    "SSL_SERVER_CERT_DN" : true,
    "SSL_SERVER_DN_MATCH" : true,
    "SSL_VERSION" : true,
    "SUPER_SHARDING_KEY" : true,
    "TRANSPORT_CONNECT_TIMEOUT" : true,
    "BACKUP" : true,
    "TYPE" : true,
    "METHOD" : true,
    "TRANSACTION" : true,
    "RETRIES" : true,
    "DELAY" : true,
    "COMMIT_OUTCOME" = true,
    "MY_WALLET_LOCATION" : false,
    "OCI_COMPARTMENT" : false,
    "OCI_DATABASE" : false,
    "OCI_PROFILE " : false,
    "OCI_TENANCY " : false,
    "TUNNEL_SERVICE_NAME" : false,
    "TENANT_ID" : false,
    "TOKEN_AUTH " : false,
    "TOKEN_LOCATION" : false,
    "WALLET_LOCATION" : false
  }
}
```

Code Sample

```
using System;
using Oracle.ManagedDataAccess.Client;

class Example
{
    static void Main()
    {

        string allowedParameters = "{
        \"DataSource\" : {
            \"CLIENT_ID\" : true,
            \"COLOCATION_TAG\" : true,
            \"CONNECT_TIMEOUT\" : true,
            \"CONNECTION_ID_PREFIX\" : true,
            \"ENABLE\" : true,
            \"EXPIRE_TIME\" : true,
            \"FAILOVER\" : true,
            \"HOST\" : true,
            \"HTTPS_PROXY\" : true,
            \"HTTPS_PROXY_PORT\" : true,
            \"IGNORE_ANO_ENCRYPTION_FOR_TCPS\" : true,
            \"INSTANCE_NAME\" : true,
            \"LOAD_BALANCE\" : true,
            \"PASSWORD_AUTH \" : true,
            \"POOL_BOUNDARY\" : true,
            \"POOL_CONNECTION_CLASS\" : true,
            \"POOL_NAME\" : true,
            \"POOL_PURITY\" : true,
            \"PORT\" : true,
            \"PROTOCOL\" : true,
            \"RECV_BUF_SIZE\" : true,
            \"RECV_TIMEOUT\" : true,
            \"RETRY_COUNT\" : true,
            \"RETRY_DELAY\" : true,
            \"SDU\" : true,
            \"SEND_BUF_SIZE\" : true,
            \"SERVER\" : true,
            \"SERVICE_NAME\" : true,
            \"SHARDING_KEY\" : true,
            \"SHARDING_KEY_ID\" : true,
            \"SOURCE_ROUTE\" : true,
            \"SSL_CERTIFICATE_THUMBPRINT\" : true,
            \"SSL_SERVER_CERT_DN\" : true,
            \"SSL_SERVER_DN_MATCH\" : true,
            \"SSL_VERSION\" : true,
            \"SUPER_SHARDING_KEY\" : true,
            \"TRANSPORT_CONNECT_TIMEOUT\" : true,
            \"BACKUP\" : true,
            \"TYPE\" : true,
            \"METHOD\" : true,
            \"TRANSACTION\" : true,
            \"RETRIES\" : true,
            \"DELAY\" : true,
            \"COMMIT_OUTCOME\" = true,
            \"MY_WALLET_LOCATION\" : false,
            \"OCI_COMPARTMENT\" : false,
            \"OCI_DATABASE\" : false,
            \"OCI_PROFILE \" : false,
            \"OCI_TENANCY \" : false,
```

```
"TUNNEL_SERVICE_NAME" : false,  
"TENANT_ID" : false,  
"TOKEN_AUTH " : false,  
"TOKEN_LOCATION" : false,  
"WALLET_LOCATION" : false  
}  
}";  
OracleConfiguration.EnableConnectionStringAllowedProperties(allowedParameters,  
enforceOnAlias: false);  
// Supply the connection string and open a connection  
OracleConnection con = new OracleConnection(<connection string>);  
con.Open();  
}  
}
```

Connection Configuration Usage and Validation

For pooled connections, allowed properties validation occurs only once during each pool's first connection creation. Changing the JSON settings will not change the existing pool settings unless either `ClearPool()` or `ClearAllPools()` is invoked. Then, the next connection open will use the new settings.

Newly created pools use the latest updated JSON settings.

In non-pooled connection scenarios, validation executes for each connection request with the latest supplied allowed properties version.

Configuring for Entity Framework Code First

Developers must configure applications to use the Oracle Entity Framework functionality. This consists of creating two entries in the `app.config` or `web.config` file and adding an assembly reference:

- Add entries in the .NET config file
 - Connection string

A standard ADO.NET connection string is used rather than the Entity Framework connection string used by Database First or Model First paths. The connection string name should match the application context name. The connection string entry is an element of the `connectionStrings` section in the configuration file.
 - Provider registration

Entity Framework uses the provider registration to determine the assembly to use for Oracle Entity Framework functionality. The provider registration is an element of the `providers` section within the `entityFramework` section in the application configuration file.
- Add Assembly reference

Add Oracle Entity Framework assembly to the project references.

Note

When using the official ODP.NET, NuGet installation, these preceding sections are created automatically, if they do not already exist. After the NuGet install, the ODP.NET connection string will need to be customized to the application's specific settings.

When using the Oracle Universal Installer or xcopy install, the preceding sections must all be configured manually.

Examples of connection strings are as follows:

- ODP.NET, Unmanaged Driver

```
<add name="TestContext" providerName="Oracle.DataAccess.Client"
connectionString="User Id=test;Password=testpassword;Data Source=eftest" />
```

- ODP.NET, Managed Driver

```
<add name="TestContext" providerName="Oracle.ManagedDataAccess.Client"
connectionString="User Id=test;Password=testpassword;Data Source=eftest" />
```

Examples of Oracle provider registration are as follows:

- ODP.NET, Unmanaged Driver

```
<provider invariantName="Oracle.DataAccess.Client"
type="Oracle.DataAccess.EntityFramework.EFOracleProviderServices,
Oracle.DataAccess.EntityFramework, Version=6.121.2.0, Culture=neutral,
PublicKeyToken=89b483f429c47342" />
```

- ODP.NET, Managed Driver

```
<provider invariantName="Oracle.ManagedDataAccess.Client"
type="Oracle.ManagedDataAccess.EntityFramework.EFOracleProviderServices,
Oracle.ManagedDataAccess.EntityFramework, Version=6.121.2.0, Culture=neutral,
PublicKeyToken=89b483f429c47342" />
```

Entity Framework 6 Code-Based Registration

Entity Framework 6 allows an application to register with an Entity Framework provider without using any configuration file. With ODP.NET, Managed Driver, the code will look as follows:

```
// C#
using Oracle.ManagedDataAccess.EntityFramework;
...
public class ModelConfiguration : DbConfiguration
{
    public ModelConfiguration()
    {
        SetProviderServices("Oracle.ManagedDataAccess.Client",
EFOracleProviderServices.Instance);
    }
}
```

For ODP.NET, Unmanaged Driver, replace occurrences of ManagedDataAccess with DataAccess in the preceding code.

If you are using code-based registration, then the configuration file should not include the registration. The configuration file based registration overrides the code-based registration.

Migrating from ODP.NET, Unmanaged Driver to ODP.NET, Managed Driver or ODP.NET Core

To take advantage of new and better feature sets in managed ODP.NET and ODP.NET Core, developers are migrating their existing unmanaged ODP.NET code to these providers. Migration from unmanaged to either managed or core ODP.NET is often straightforward and simple.

The APIs of ODP.NET, Managed Driver and ODP.NET Core are nearly identical. Those two provider APIs generally match ODP.NET, Unmanaged Driver APIs.

As unmanaged ODP.NET is deprecated, it will see fewer new features in the future. Managed and core providers will continually add new Oracle database, cloud, and .NET feature support.

To migrate from unmanaged ODP.NET to managed ODP.NET or ODP.NET Core, perform the following steps:

1. Add a Reference to `Oracle.ManagedDataAccess.dll` in the .NET project.
2. Change the existing ODP.NET, Unmanaged Driver namespace references to ODP.NET, Managed Driver references.

```
// C#  
using Oracle.ManagedDataAccess.Client;  
using Oracle.ManagedDataAccess.Types;  
  
// VB  
Imports Oracle.ManagedDataAccess.Client  
Imports Oracle.ManagedDataAccess.Types
```

3. Some provider configuration settings may need to be migrated because ODP.NET, Managed Driver supports very few Windows Registry settings and a different .NET configuration setting format. ODP.NET Core does not support Windows Registry settings nor .NET Framework configuration files, that is, `app/machine/web.config`.

① See Also

[Configuring Oracle Data Provider for .NET](#) for more information.

Configuring a Port to Listen for Database Notifications

Oracle Data Provider for .NET opens a port to listen for database notifications when the following features are used:

- HA Events
- Load Balancing
- Continuous Query Notification
- AQ Notifications

All these features share the same port, which can be configured centrally by setting the db notifications port in an application or web configuration file.

If the configuration file does not exist or the db notification port is not specified, ODP.NET uses a valid, random port number. The configuration file may also request for a random port by specifying a db notification port value of -1. To specify a particular port in ODP.NET, Unmanaged Driver, for example, 1200, an application or web configuration file can be used as follows:

```
<configuration>
  <oracle.dataaccess.client>
    <settings>
      <add name="DbNotificationPort" value="1200"/>
    </settings>
  </oracle.dataaccess.client>
</configuration>
```

To specify a particular port in ODP.NET, Managed Driver, an application or web configuration file can be used as follows:

```
<configuration>
  <oracle.manageddataaccess.client>
    <version number="*">
      <settings>
        <setting name="DbNotificationPort" value="1200"/>
      </settings>
    </version>
  </oracle.manageddataaccess.client>
</configuration>
```

The port number should be unique for each process running on a computer. Thus, the port number should be set uniquely for each application either programmatically or through an application config file. Note that if the specified port number is already in use or invalid, ODP.NET does not provide any errors.

When the process using ODP.NET starts, the application reads the db notification port number and listens on that port. Once the port is opened, the port number cannot be changed during the lifetime of the process.

See Also

- ["Fast Application Notification"](#)
- ["Runtime Connection Load Balancing"](#)
- ["Continuous Query Notification Support "](#)
- ["Advanced Queuing and Transactional Event Queues"](#)

General .NET Programming Recommendations and Tips for ODP.NET

- `Thread.Abort()` should not be used, as unmanaged resources may remain unreleased, which can potentially cause memory leaks and hangs.
- To optimize resource usage, ODP.NET objects, such as `OracleConnection` and `OracleCommand`, should be explicitly closed or disposed, or both, when they are no longer needed. This should be done rather than relying on the .NET Framework garbage collector

to reclaim resources. Many users have found that under stress conditions, explicit `Close` or `Dispose` calls result in much lower resource usage.

- It is recommended not to proceed with application execution if the application encounters exceptions that are associated with possible memory corruption, such as `System.AccessViolationException` and `System.Runtime.InteropServices.SEHException`.
- If the `HKEY_LOCAL_MACHINE\Software\Oracle\NLS_LANG` registry entry is set to `NA`, ODP.NET encounters `ORA-12705` errors. To eliminate this problem, remove the `HKEY_LOCAL_MACHINE\Software\Oracle\NLS_LANG` registry entry.
- .NET requires certain special characters, such as backslash `\` and double quotation mark `"`, to be escaped when used within a string. In a connection string or a class property, .NET requires using the escape character, backslash, before the special character. To represent a backslash in a string, use a double backslash `\\`. To represent a double quotation mark in a string, use a backslash followed by a double quotation mark `\"`. Alternatively, use the at sign `@` to mark the entire string as a verbatim string literal, in which case .NET ignores special characters.
- If your application does not use LDAP, then it is not necessary to include the `System.DirectoryServices` and `System.DirectoryServices.Protocols` libraries in your application.

3

Features of Oracle Data Provider for .NET

This section describes Oracle Data Provider for .NET provider-specific features and how to use them to develop .NET applications.

This section contains the following topics:

- [Base Classes, Provider Factory Classes, and DbDataSource](#)
- [Code Access Security](#)
- [Connecting to Oracle Database](#)
- [Real Application Clusters and Global Data Services](#)
- [Using Transaction Guard to Prevent Logical Corruption](#)
- [Application Continuity](#)
- [Transparent Application Failover](#)
- [Oracle Globally Distributed Database](#)
- [OracleCommand Object](#)
- [ODP.NET Types Overview](#)
- [GUIDs](#)
- [Obtaining Data from an OracleDataReader Object](#)
- [PL/SQL REF CURSOR and OracleRefCursor](#)
- [Implicit REF CURSOR Binding](#)
- [LOB Support](#)
- [Native JSON Support](#)
- [Artificial Intelligence Vectors and Semantic Search](#)
- [ODP.NET XML Support](#)
- [Oracle User-Defined Types \(UDTs\) and .NET Custom Types](#)
- [Bulk Copy](#)
- [Asynchronous Programming and Pipelining](#)
- [Advanced Queuing and Transactional Event Queues](#)
- [Continuous Query Notification Support](#)
- [OracleDataAdapter Safe Type Mapping](#)
- [OracleDataAdapter Requery Property](#)
- [Guaranteeing Uniqueness in Updating DataSet to Database](#)
- [Globalization Support](#)
- [Data Use Case Domains](#)
- [Annotations](#)
- [OpenTelemetry](#)

- [.NET Metrics](#)
- [Debug Tracing](#)
- [Database Application Migration: SQL Translation Framework](#)
- [Allowed Logon Version Client Support](#)

Base Classes, Provider Factory Classes, and DbDataSource

Base Classes and Provider Factory Classes

With ADO.NET, data classes derive from the base classes defined in the System.Data.Common namespace. Developers can create provider-specific instances of these base classes using provider factory classes.

Provider factory classes allow generic data access code to access multiple data sources with a minimum of data source-specific code. This reduces much of the conditional logic currently used by applications accessing multiple data sources.

Using Oracle Data Provider for .NET, the OracleClientFactory class can be returned and instantiated, enabling an application to create instances of the following ODP.NET classes that inherit from the base classes:

Table 3-1 ODP.NET Classes that Inherit from ADO.NET Base Classes

ODP.NET Classes	Inherited from ADO.NET Base Class
OracleClientFactory	DbProviderFactory
OracleCommand	DbCommand
OracleCommandBuilder	DbCommandBuilder
OracleConnection	DbConnection
OracleConnectionStringBuilder	DbConnectionStringBuilder
OracleDataAdapter	DbDataAdapter
OracleDataReader	DbDataReader
OracleDataSourceEnumerator	DbDataSourceEnumerator
OracleException	DbException
OracleParameter	DbParameter
OracleParameterCollection	DbParameterCollection
OracleTransaction	DbTransaction

In general, applications still require Oracle-specific connection strings, SQL or stored procedure calls, and declare that a factory from ODP.NET is used.

 **See Also**
[OracleClientFactory Class](#)

DbDataSource

ODP.NET Core supports the ADO.NET DbDataSource abstraction through the `OracleDataSource` class. The new `OracleDataSource` class can retrieve connections and execute commands directly as an alternative to using `OracleConnection` and `OracleCommand`.

Since the class encapsulates all Oracle database connection configuration, `OracleDataSource` is portable and can be registered in dependency injection as a connection factory. It is ideal when provider agnostic code is preferred when connecting to Oracle databases.

`OracleDataSource` does not have a constructor; it must be created using the `OracleDataSourceBuilder` class or through the `OracleClientFactory.CreateDataSource(string connectionString)` method.

In ODP.NET, many connection configuration options are not available on the connection string. For those configuration options, developers can use `OracleDataSourceBuilder`, then build an `OracleDataSource` from it.

ODP.NET Core 23.8 began support for `OracleDataSource` and `OracleDataSourceBuilder` with .NET 8 runtime or higher.

① See Also

[OracleDataSource Class](#)

[OracleDataSourceBuilder Class](#)

[OracleDataSourceCommand Class](#)

OracleDataSource Connections

`OracleDataSource` can return open or closed connections synchronously or asynchronously. It can also create commands and batch operations that use the connection string.

`OracleDataSource` Code Sample:

```
OracleClientFactory factory = new OracleClientFactory();

// Create OracleDataSource
DbDataSource oraDS = factory.CreateDataSource("user id=hr;password=<PASSWORD>;data
source=oracle");

// Create closed connection, which uses the connection string set on OracleDataSource
DbConnection connection = oraDS.CreateConnection();
connection.Open();

connection.Close();

// An OracleCommand can be created directly from the OracleDataSource, which will be
ready to execute commands
DbCommand cmd = oraDS.CreateCommand("select * from employees");
DbDataReader rdr = cmd.ExecuteReader();
while (rdr.Read())
{
    Console.WriteLine(rdr.GetString(1));
}
rdr.Close();
```

When pooling is enabled, each `OracleDataSource` instance has its own corresponding connection pool, allowing connections to be dispensed faster by skipping the look up process. Because `OracleDataSource` corresponds to a specific connection pool, an `OracleDataSource` is immutable once created. However, `OracleConnections` created from an `OracleDataSource` are mutable.

`OracleDataSource` connections use a different pool from traditional `OracleConnection` pools even when they have the same connection properties. By the same principle, `OracleConnection.ClearAllPools()` does not affect pools maintained by `OracleDataSource` objects. `OracleDataSource` has its own `ClearPool()` method. When an `OracleDataSource` is disposed, its corresponding pool is cleared.

After an `OracleConnection` is created from an `OracleDataSource`, its properties can be changed when the connection is closed. However, some connection properties are used to assign the pool a unique identifier. Modifying one or more of those properties removes the connection from the existing `OracleDataSource` pool and either creates a new pool or uses another existing pool.

Passwords can be updated through the `OracleDataSource.Password` and `OracleDataSource.ProxyPassword` properties. Upon the next connection open, the new password will be used to authenticate the user. These properties do not update the database password. Passwords are the only mutable `OracleDataSource` property.

OracleDataSource Performance Counters

Pool names in ODP.NET performance counters are prefixed with "ODS: " for `OracleDataSource` connections, such as "ODS: user id=HR; data source=oracle".

Two `OracleDataSource` instances with the same connection string will have separate pools. Oracle recommends setting pool name values to distinguish each uniquely. The pool name can be set via `OracleConnection.PoolName` or `OracleDataSourceBuilder.UsePoolName`.

Developers are responsible for ensuring there each pool has a unique pool name.

The pool name must be set before the pool is created. ODP.NET will not use the pool name value after the pool is created nor will it throw an error.

OracleDataSource Connection Event Callbacks

Just like `OracleConnection` properties, `OracleDataSource` and `OracleDataSourceBuilder` can employ event callbacks. Their event behavior is near equivalent to `OracleConnection`'s except for `HAEvent`. `OracleConnection.HAEvent` is static, which means it registers events for all `OracleConnection` instances. `OracleDataSource.HAEvent` is not static; it is instance based like the other events. `HAEvent` registered on the `OracleDataSourceBuilder` only affects connections created from the `OracleDataSource`, not all connections.

This table shows the `OracleConnection`, `OracleDataSource`, and `OracleDataSourceBuilder` event properties and their common or unique behavior.

OracleConnection Property	OracleDataSource Property	OracleDataSource Method	Behavior Description
ConnectionOpen InfoMessage StateChange	ConnectionOpen InfoMessage StateChange	UseConnectionOpenCallback UseInfoMessageCallback UseStateChangeCallback	Applies event callback to all OracleConnection instances created by OracleDataSource, at which point it behaves identically to setting the event on OracleConnection itself.
Failover	Failover	UseFailoverCallback	Behavior is almost identical, except OracleConnection.Failover can only be set if the OracleConnection is open. OracleDataSourceBuilder.Failover applies the event to all OracleConnections created from the OracleDataSource even if they are initially closed.
HAEvent	N/A	N/A	Being static, it registers callbacks for all OracleConnection instances.
N/A	HAEvent	UseHAEventCallback	Being instance-based, it only registers callbacks for OracleConnection instances created by OracleDataSource.

OracleDataSource Commands

`OracleDataSource.CreateCommand()` returns a new `OracleDataSourceCommand` class, which is derived from `OracleCommand`. The class functions almost identically to `OracleCommand` with a few differences.

First, the connection object is not accessible to the app. `OracleDataSourceCommand` internally opens a connection before an execution call and closes it after the completing the execution except for `ExecuteReader`. The reader depends on the connection to retrieve results. The connection closes when the reader closes.

Applications typically defer retrieving all their ODP.NET connected types, such as LOB and LONG data, which requires the connection to remain open to be able to fetch data in the future. When using connected types with `OracleDataSourceCommand` and without `ExecuteReader`, set the `InitialLOBFetchSize` or `InitialLONGFetchSize` property to a sufficiently large value to retrieve all the result set LOB and LONG data immediately. If any LOB or LONG data remains and the app attempts to fetch it, an exception is thrown.

Note

OracleDataSourceCommand is not compatible with OracleDataAdapter.

OracleDataSourceCommand can only be created through `OracleDataSource.CreateCommand()`.

Since connection objects are not accessible to `OracleDataSource`, the `OracleDataSourceCommand.Connection` and `OracleDataSourceCommand.Transaction` properties throw an `InvalidOperationException` when accessed.

ExecuteNonQuery() example

```
OracleDataSource ds = new OracleDataSourceBuilder("<CONNECTION STRING>").Build();
OracleCommand cmd = ds.CreateCommand("INSERT INTO mytable VALUES(0, 1)");
// ODP.NET internally opens a connection when ExecuteNonQuery is called
OracleDataReader rdr = cmd.ExecuteNonQuery();
// ODP.NET closes the connection after ExecuteNonQuery completes.
```

ExecuteReader() example

```
OracleDataSource ds = new OracleDataSourceBuilder("<CONNECTION STRING>").Build();
OracleCommand cmd = ds.CreateCommand("SELECT * FROM mytable");
// ODP.NET internally opens a connection when ExecuteReader is called
OracleDataReader rdr = cmd.ExecuteReader();
while (rdr.Read())
{
    Console.WriteLine(rdr.GetString(0));
}
rdr.Close();
// ODP.NET closes the connection when the reader is closed.
```

Code Access Security

ODP.NET implements code access security through the `OraclePermission` class. This ensures that application code trying to access the database has the requisite permission to do so.

When a .NET assembly tries to access Oracle Database through ODP.NET, ODP.NET demands `OraclePermission`. The .NET runtime security system checks to see whether the calling assembly, and all other assemblies in the call stack, have `OraclePermission` granted to them. If all assemblies in the call stack have `OraclePermission` granted to them, then the calling assembly can access the database. If any one of the assemblies in the call stack does not have `OraclePermission` granted to it, then a security exception is thrown.

Note

ODP.NET Core does not support Code Access Security.

Configuring OraclePermission

The DemandOraclePermission configuration attribute is used to enable or disable OraclePermission demand for an ODP.NET API. The DemandOraclePermission value can be specified in the Windows registry for unmanaged ODP.NET only, or an individual application configuration file for both unmanaged and managed ODP.NET.

The following Windows registry key is used to configure the DemandOraclePermission configuration attribute:

```
HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\ODP.NET\Assembly_Version\DemandOraclePermission
```

Here *Assembly_Version* is the full assembly version number of Oracle.DataAccess.dll. The DemandOraclePermission key is of type REG_SZ. It can be set to either 1 (enabled) or 0 (disabled).

You can also enable OraclePermission demand for an individual application using its application configuration file. The following example enables the DemandOraclePermission property in an application configuration file for ODP.NET, Unmanaged Driver:

```
<configuration>
  <oracle.dataaccess.client>
    <settings>
      <add name="DemandOraclePermission" value="1"/>
    </settings>
  </oracle.dataaccess.client>
</configuration>
```

Similarly, you can use DemandOraclePermission to configure ODP.NET, Managed Driver under the settings section for managed provider configuration. See also "[settings section](#)" for more information.

An application or assembly can successfully access the database if OraclePermission has been added to the permission set associated with the assembly's code group. A system administrator can modify the appropriate permission set manually or by using the Microsoft .NET configuration tool (Mscorcfg.msc).

Administrators may also use an appropriate .NET Framework Tool, such as the Code Access Security Policy Tool (Caspol.exe), to modify security policy at the machine, user, and enterprise levels for including OraclePermission.

OracleConnection makes security demands using the OraclePermission object when OraclePermission demand has been enabled using DemandOraclePermission configuration attribute. Application developers should make sure that their code has sufficient permission before using OracleConnection.

See Also

- "[OraclePermission Class](#)"
- "[OraclePermissionAttribute Class](#)"

Configuring OraclePermission for Web Applications with High or Medium Trust Levels

For Web applications operating under high or medium trust, OraclePermission needs to be configured in the appropriate `web_TrustLevel.config` file, so that the application does not encounter any security errors.

OraclePermission can be configured using the OraProvCfg tool. OraProvCfg.exe adds appropriate entries to the `web_hightrust.config` and `web_mediumtrust.config` files associated with the specified .NET framework version. The following example illustrates using the OraProvCfg tool for configuring OraclePermission in a .NET 2.0 Web application:

```
OraProvCfg.exe /action:config /product:odp /component:oraclepermission  
               /frameworkversion:v2.0.50727  
               /providerpath:full_path_of_Oracle.DataAccess.dll
```

On running the preceding command, the following entry is added to the `web_hightrust.config` and `web_mediumtrust.config` files under the ASP.NET permission set:

```
<IPermission class="Oracle.DataAccess.Client.OraclePermission, Oracle.DataAccess,  
Version=2.112.2.0, Culture=neutral, PublicKeyToken=89b483f429c47342" version="1"  
Unrestricted="true" />
```

OraProvCfg can also be used to remove these entries from the .config files when required. The following example illustrates this:

```
OraProvCfg.exe /action:unconfig /product:odp /component:oraclepermission  
               /frameworkversion:v2.0.50727  
               /providerpath:full_path_of_Oracle.DataAccess.dll
```

Configuring OraclePermission for Windows Applications Running in a Partial Trust Environment

For Windows applications operating in a partial trust environment, the OraclePermission entry should be specified under the appropriate permission set in the `security.config` file. The `security.config` file is available in the `%windir%\Microsoft.NET\Framework\{version}\CONFIG` folder.

The following example specifies the OraclePermission entry for a .NET 2.0 Windows application:

```
<IPermission class="Oracle.DataAccess.Client.OraclePermission, Oracle.DataAccess,  
Version=2.112.2.0, Culture=neutral, PublicKeyToken=89b483f429c47342" version="1"  
Unrestricted="true" />
```

Connecting to Oracle Database

Oracle Data Provider for .NET can connect to Oracle Database in a number of ways, such as using a user name and password, Windows Native Authentication, Kerberos, and Transport Layer Security/Secure Sockets Layer. This section describes OracleConnection provider-specific features, including:

- [Connecting to Oracle Autonomous Database](#)
- [Using Oracle Identity and Access Management](#)

- [Using Microsoft Entra ID](#)
- [Connection String Attributes and URLs](#)
- [Connection String Builder](#)
- [Specifying the Data Source Attribute](#)
- [Using WebSocket](#)
- [Using Transport Layer Security and Secure Sockets Layer](#)
- [Using Secure External Password Store](#)
- [Using Kerberos](#)
- [Using Windows Native Authentication \(NTS\)](#)
- [Network Data Encryption and Integrity](#)
- [Schema Discovery](#)
- [Connection Pooling](#)
- [Connection Pool Management](#)
- [Connection Performance Counters](#)
- [Database Resident Connection Pooling and Connection Manager in Traffic Director Mode](#)
- [Oracle Multitenant and Pluggable Databases](#)
- [Edition-Based Redefinition](#)
- [Privileged Connections](#)
- [Connection Pooling with OracleCredential](#)
- [Password Expiration](#)
- [Proxy Authentication](#)
- [Dynamic Distributed Transaction Enlistment](#)
- [Client Identifier and End-to-End Tracing](#)
- [Automatic Connection Recovery](#)
- [Reset Session State to Limit Application State Leaks](#)

Connecting to Oracle Autonomous Database

Oracle Autonomous Database includes several deployment options:

- Oracle Autonomous Database Serverless (ADB-S)
- Oracle Autonomous Database on Dedicated Exadata Infrastructure (ADB-D)
- Oracle Autonomous Database on Exadata Cloud@Customer (ADBC@C)

TLS/SSL must be used to connect to ADB-S. For ADB-D and ADBC@C, use of TLS is optional. Currently, only ADB-S can be used with Oracle Identity and Access Management.

This documentation section focuses on how to connect to ADB-S.

Unmanaged ODP.NET 19.14 and 21.5 and higher supports TLS connections without a wallet.

ODP.NET core and managed drivers 19.13 and 21.4 and higher support TLS connections without a wallet.

ODP.NET Core 19.13.1 or 21.4.1 and higher support token based authentication with Oracle Identity and Access Management (IAM).

Managed ODP.NET 21.6.1 now supports token based authentication with IAM.

Oracle recommends using the latest ODP.NET driver version available on NuGet Gallery or Oracle website when connecting to Oracle ADB-S.

Connecting with a Wallet

The following quick start link has instructions how to download, install, and configure ODP.NET and Oracle Developer Tools for Visual Studio when using a wallet:

[Quick Start: Developing .NET Applications for Oracle Autonomous Database](#)

Connecting without a Wallet

When you connect to Oracle ADB-S with ODP.NET, you do not need to deploy the Oracle wallet or the Oracle network configuration files `sqlnet.ora` or `tnsnames.ora` with your application. Instead, you supply the data source attribute, a TLS connection string, with the configuration information in the ODP.NET connection.

To use ODP.NET TLS connections, do the following:

1. Obtain managed ODP.NET or ODP.NET Core versions 19.13 or 21.4 or above. For unmanaged ODP.NET, versions 19.14 and 21.5 can be obtained. Lower level versions do not support TLS connections without wallets.
2. Enable TLS connections on your Autonomous Database instance. See [Update your Autonomous Database Instance to Allow both TLS and mTLS Authentication](#) for details.
3. After you enable TLS connections, supply a TLS connection string in the ODP.NET data source to connect to your Autonomous Database instance. See [View TNS Names and Connection Strings for an Autonomous Database Instance](#) for details on viewing or copying TLS connection strings.

Note

Older TNS names use a distinguished name and place quotation marks around it. Newer TNS names do not have it. If you store one of these older TNS names in a .NET string, add a backslash escape sequence before each quotation mark (for example, `\"`). This allows .NET to recognize the quotation mark as part of the TNS name.

See Also

- [Tips for Connecting .NET Apps to Oracle Autonomous Database](#)
- [Autonomous Database Cloud Services Documentation](#)
- [Oracle Database Cloud Services Documentation](#)

Using Oracle Identity and Access Management

Oracle identity and access management (IAM) solutions provide secure access to enterprise applications for both cloud and on-premises deployments. Oracle IAM cloud service supports a unified identity across Oracle cloud services, including Oracle ADB-S. This capability allows identity to be propagated to all services Oracle IAM supports. A unified identity makes user management and account management easier for administrators and end users.

IAM users can connect to the Oracle Autonomous Database instance by using either an IAM database password verifier, an IAM token, or a database alternate password for token authentication.

Using the IAM database password verifier is similar to the Oracle Database password authentication process. However, instead of the password verifier (encrypted hash of the password) being stored in the Oracle Autonomous Database, the verifier is instead stored as part of the Oracle Cloud Infrastructure (OCI) IAM user attributes. This option is supported by all ODP.NET clients.

The second connection method, the use of an IAM token for the database, is more modern. The use of token-based access is a better fit for Cloud resources such as Oracle Autonomous Database. The token is based on the strength that the IAM endpoint can enforce. This can be multi-factor authentication, which is stronger than the use of passwords alone. Starting with versions 19.13.1 and 21.4.1, ODP.NET Core supports this new connection method. This functionality was added to managed ODP.NET in version 21.6.1. This will utilize native OCI Authentication primitive, specifically OCI supported tokens for authenticating OCI users when they logon to a database. By integrating with OCI IAM for authentication, Oracle databases in OCI will provide seamless identity integration with OCI services.

The third connection method is the database password for IAM token authentication. This method is different from using the IAM database password verifier since it is using a database alternate password for token authentication. This connection method is also called token-based authentication using password. ODP.NET Core and Managed ODP.NET support this feature starting with version 21.6.1.

The fourth connection method is IAM single sign-on token (SSO) authentication. Starting with ODP.NET 23ai, Oracle IAM single sign-on is supported with the managed and core provider types. ODP.NET will automatically handle IAM token retrieval and management. Oracle Database 19c (19.16), Oracle AI Database 26ai, and higher versions support IAM SSO.

① See Also

See also [Use Identity and Access Management \(IAM\) Authentication with Autonomous Database in Using Oracle Autonomous Database Serverless](#).

Configuring a Client Connection Using an IAM Database Password Verifier

After you have configured the authorization needed for the IAM user, this user can log in without additional configuration.

The application provides the IAM user name and IAM database password, not the Oracle Cloud Infrastructure (OCI) console password, using any ODP.NET driver type. The only constraint is that the ODP.NET driver must have support for Oracle Database 12c passwords verifiers. Earlier password verifiers are not supported with IAM. No special client or tool

configuration is needed for the IAM user to connect to the Oracle Autonomous Database instance.

Configuring a Client Connection Using Token

ODP.NET supports the IAM token, which is a Signature token represented in standard JSON Web Token (JWT). Signature Token – also called Proof of Possession (PoP) token – is associated with a private key known to the client. The client sends a signature and token to the protected resource. The protected resource verifies the signature along with the token itself.

For passing IAM token to Oracle Autonomous Database, the following options are supported:

- Use of token file: For existing applications, use Oracle Cloud Infrastructure (OCI) command line interface (CLI) to request and store the database token in the file system, and configure ODP.NET to use the token file.
- Use of API: Modify application to request database token from IAM and pass the database token to the database client through ODP.NET API.

ODP.NET presents the database token when connecting to a database. The database requests the public key from IAM and the database validates the database token. If the token is valid, the database requests the IAM groups that the user is a member. The database reviews the local schema mappings to find which global schema the IAM user will have access to, exclusive or shared. If no schema mapping exists, the user will not be allowed to access the database.

Configuring Client Connection to use Token File

The OCI CLI is available to request database tokens from IAM. When the OCI CLI tool is used to request tokens, it copies the token and private key to the default directory on the local machine or a custom directory can be specified. On Windows operating systems, the default directory is based on the USERPROFILE environment variable (i.e. `$USERPROFILE/.oci/db-token`). On Linux and macOS operating systems, the default directory is based on the HOME environment variable (i.e. `$HOME/.oci/db-token`).

To use token file for authentication:

1. Set ODP.NET User Id to "/" in the connection string and leave the password empty.
2. Set `TOKEN_AUTH` parameter to `OCI_TOKEN`. This parameter can be configured at various levels like TNS descriptor, EZConnect string, `sqlnet.ora` file or application configuration file. This can also be configured programmatically either at connection level through the `OracleConnection.TokenAuthentication` property or at process level through the `OracleConfiguration.TokenAuthentication` property.
3. If using a non-default token location, set the token file directory. More details on setting a specific token file location is discussed in the "Use of ODP.NET API to pass the Token" section.

When database token usage is enabled, all other external authentication methods, such as Kerberos, Windows authentication, and TLS cannot be used. However, TLS to encrypt the client-server connection can be used – and must be used when using IAM tokens.

① See Also

- Authenticating and Authorizing IAM Users for Oracle Autonomous Databases for all the steps needed to get the token file.
- [OCI Command Line Interface](#)

Use of ODP.NET API to pass the Token

Applications can use OCI SDK to get database token for IAM user and pass the database token along with private key to ODP.NET through the API.

The database token has attributes within the token, some of which are worth noting, which will be covered in more detail later:

- exp attribute within the token represents the expiry time of the token.
- sub attribute within the token represents the IAM user of the token.

The `OracleConnection.AccessToken` property can be set to the `OracleAccessToken` object. The `OracleAccessToken` object can be constructed by providing both the signature token and private key, which are required to connect via IAM.

If `OracleConnection.AccessToken` property is not supplied by the application and the `TOKEN_AUTH` parameter is set to `OCI_TOKEN`, then ODP.NET will look for the signature token and private key files in the default directory. Applications can override this default directory by setting the `TOKEN_LOCATION` parameter. This parameter can also be configured at various levels like TNS descriptor, EZConnect string, `sqlnet.ora` file, or application configuration file. This parameter can also be configured programmatically either at the ODP.NET connection level through the `OracleConnection.TokenLocation` property or at process level through the `OracleConfiguration.TokenLocation` property.

The following table shows how ODP.NET will use IAM when a signature token is supplied and when `TOKEN_AUTH` / `TokenAuthentication` property is set.

Table 3-2 ODP.NET Authentication using IAM and Signature Token

Application Supplied Oracle Access Token	TOKEN_AUTH / TokenAuthentication Setting	PASSWORD_AUTH / PasswordAuthentatio n Setting	ODP.NET Authentication
Yes	OCI_TOKEN / <code>OracleTokenAuth.Oci Token</code>	NOT SET / <code>OraclePasswordAuth. PasswordVerifier</code> (Default)	IAM via application SUPPLIED TOKEN
Yes	NOT SET / <code>OracleTokenAuth.Dis abled (default)</code>	NOT SET / <code>OraclePasswordAuth. PasswordVerifier</code> (default)	IAM via application SUPPLIED TOKEN
No (default)	OCI_TOKEN / <code>OracleTokenAuth.Oci Token</code>	NOT SET / <code>OraclePasswordAuth. PasswordVerifier</code> (Default)	IAM via file SUPPLIED TOKEN

Table 3-2 (Cont.) ODP.NET Authentication using IAM and Signature Token

Application Supplied Oracle Access Token	TOKEN_AUTH / TokenAuthentication Setting	PASSWORD_AUTH / PasswordAuthentatio n Setting	ODP.NET Authentication
No (default)	NOT SET / OracleTokenAuth.Dis abled (default)	NOT SET / OraclePasswordAuth. PasswordVerifier (Default)	External Authentication, such as Kerberos, Windows, and TLS/SSL
Yes	OCI_TOKEN / OracleTokenAuth.Oci Token	OCI_TOKEN / OraclePasswordAuth. OciToken	IAM via application SUPPLIED TOKEN
Yes	NOT SET / OracleTokenAuth.Dis abled (default)	OCI_TOKEN / OraclePasswordAuth. OciToken	IAM via application SUPPLIED TOKEN
No (default)	OCI_TOKEN / OracleTokenAuth.Oci Token	OCI_TOKEN / OraclePasswordAuth. OciToken	If userid=/, IAM via file supplied token. Else if TokenAuthentication set programmatically, then error. In all other cases, token from REST service.
No (default)	NOT SET / OracleTokenAuth.Dis abled (default)	OCI_TOKEN / OraclePasswordAuth. OciToken	Token From REST Service if userid/ password set or userid= / with SEPS enabled. In all other cases, External Authentication.

ODP.NET applications using the API to provide signature token and private key are recommended to register with the RefreshAccessToken event exposed on the OracleAccessToken object to receive a callback when token is about to expire. The .NET event provides a convenient way for applications to “refresh” the signature token and the private key before they expire.

ODP.NET triggers the token refresh callback 60 seconds before it expires. It is the application's responsibility to ensure that ODP.NET always has a valid token for authentication when creating new connections to the database. Please note that the callback will be called only once for an authenticated token.

After the callback provides the refreshed token and private key, all new connections start using the refreshed token for authentication. However, the token expiry has no impact on existing connections checked out from or within the connection pool. If for some reason, applications fail to provide a valid refreshed token and private key through the callback, it can use OracleConnection object's OpenWithNewToken() method to open a connection by passing a new signature token and private key.

It's important to note that already established idle connections will be dispensed even when the token has expired. The newly supplied signature token will be used only to create new connections. If the supplied signature token is invalid or expired, applications will receive an error during the Open() method invocation on the OracleConnection object.

IAM Token Functionality Requirements

With ODP.NET Core NuGet packages, applications must target .NET 5.0 or higher to use IAM tokens.

Managed ODP.NET does not have any unique .NET assembly version dependencies to use IAM tokens.

Sample Code

Here is ODP.NET sample code showing how to use `OracleAccessToken` to establish a connection and how to propagate the refreshed token through the `OracleRefreshAccessTokenEventArgs` class. This sample code works with managed ODP.NET and ODP.NET Core.

```
// C#
using System;
using System.IO;
using Oracle.ManagedDataAccess.Client;

class RefreshDbTokenSample
{
    // Update this path based from where token and private key would be read.
    private static string tokenPath = @"C:\token_dir";

    static void Main(string[] args)
    {
        // Create token and private key file path.
        string tokenFile = Path.Combine(tokenPath, "token");
        string privateKeyFile = Path.Combine(tokenPath, "oci_db_key.pem");

        // Read token and private key contents.
        char[] dbToken = File.OpenText(tokenFile).ReadToEnd().ToCharArray();
        char[] privateKey = File.OpenText(privateKeyFile).ReadToEnd().ToCharArray();

        // Create OracleAccessToken
        OracleAccessToken accessToken = new OracleAccessToken(dbToken, privateKey);

        // Set the token refresh call back
        accessToken.RefreshAccessToken += new
        OracleRefreshAccessTokenEventHandler(OnRefreshAccessToken);

        // establish a connection
        string constr = "user id=//;data source=oracle";

        // Create connection object
        OracleConnection con = new OracleConnection(constr);

        // Set accessToken to the connection before opening connection.
        con.AccessToken = accessToken;

        // Open connection now.
        con.Open();

        Console.WriteLine("Connected using DB Token Authentication");

        con.Dispose();
    }
}
```

```

    }

    public static void OnRefreshAccessToken(
        OracleRefreshAccessTokenEventArgs eventArgs)
    {
        // Application code to get Refreshed DB token and private key
        RefreshToken(out char[] refreshedToken, out char[] refreshedPrivateKey);

        // Set the refreshed DB token and private key to eventArgs
        eventArgs.DbToken = refreshedToken;
        eventArgs.PrivateKey = refreshedPrivateKey;
    }

    public static void RefreshToken(out char[] refreshedToken, out char[]
refreshedPrivateKey)
    {
        refreshedToken = null;
        refreshedPrivateKey = null;

        // TODO: Add code to refresh the token.
    }
}

```

Configuring a Client Connection Using Database Alternate Password for Token Authentication

The database alternate password connection method uses a username and password specific for IAM token-based authentication. This set of credentials is separate from the more common database username and password.

To use this feature, the `PASSWORD_AUTH` parameter must be set to `OCI_TOKEN`.

IAM exposes a REST endpoint for clients to request database bearer tokens. ODP.NET makes calls to request tokens from the endpoint via TLS 1.2 or higher. The driver specifies a username, password, oci iam url, tenant Oracle Cloud Identifier (OCID), a database compartment identifier (optional), and a database identifier (optional) in the request. If a database identifier is specified, then the database compartment identifier becomes mandatory. A bearer token is returned to ODP.NET in the response. The driver then sends the returned bearer token to authenticate with the database.

ODP.NET caches the token for reuse when additional connection requests occur. It holds onto the token until its expiration. The application will need to refresh the token before it expires to be able to make connections.

ODP.NET alternate password credentials can be stored securely in a Secure External Password Store (SEPS) wallet for application use.

Comparing IAM Token Authentication Methods

The following table identifies the differences and similarities among the IAM token authentication options.

App provides Database Token and Private Key	Database Token and Private Key through File	Alternate User Credentials for Token Authentication
Username must be set to "/" in the connection string.	Username must be set to "/" in the connection string.	Username and Password in ODP.NET connection string represent alternate credentials.
-	-	SEPS can store credentials. When SEPS is enabled, the username and password is retrieved first. Then, the parameter, PASSWORD_AUTH, determines whether database access occurs via username/password or token-based authentication.
Setting OracleCredential results in an error when opening a connection.	Setting OracleCredential results in an error when opening a connection.	OracleCredential can be set with alternate username and password. The username/password is provided through the connection string or OracleCredential.
No configuration parameter	TOKEN_AUTH=OCI_TOKEN	PASSWORD_AUTH=OCI_TOKEN
No configuration parameter	TOKEN_LOCATION can also be set to override default directory location of token and private file. You can set both TOKEN_AUTH and PASSWORD_AUTH to OCI_TOKEN on the same connection. In this scenario, if the username is set to "/", then file-based token authentication is used. Otherwise, alternate password is used.	OCI_IAM_URL and OCI_TENANCY must be set. OCI_COMPARTMENT and OCI_DATABASE can be set to provide specific compartment identifier and database identifier, respectively. If OCI_DATABASE is set, then, OCI_COMPARTMENT is mandatory. You can set both TOKEN_AUTH and PASSWORD_AUTH to OCI_TOKEN on the same connection. In this scenario, if the username is set to "/", then file-based token authentication is used. Otherwise, alternate password is used.
OracleAccessToken must be set on connection with proper database token and private key before opening.	-	-
Database token provided by the application must be Signature token.	Database token read from the file must be Signature token.	Database token received from IAM is bearer token.

App provides Database Token and Private Key	Database Token and Private Key through File	Alternate User Credentials for Token Authentication
ODP.NET sends the following information to the database upon opening a connection: • database token • header • Signature generated by signing the header using private key	ODP.NET reads the database token and private key from the files and sends the following information to the database for authentication upon opening a connection; • database token • header • Signature generated by signing the header using private key	Upon opening a connection, ODP.NET uses the username/ password to request the database token from IAM and then sends that token to the database for authentication.

IAM Single Sign-On

Managed ODP.NET and ODP.NET Core support Oracle IAM single sign-on (SSO), allowing user identity to be propagated across Oracle databases and cloud services, including federated users. SSO is easier for end users as they only need to log on once to access all the services that use an Oracle IAM identity. It is also easier for .NET developers as ODP.NET performs access token management automatically. The application is freed from dealing with the database token and private key.

When ODP.NET is given the Oracle IAM authentication parameters, it acquires the IAM access token to establish the database connection and all subsequent connection requests. The token is cached on a per connection pool basis. When the token is about to expire, ODP.NET automatically retrieves a new access token.

ODP.NET IAM SSO requires `Oracle.ManagedDataAccess.Oci` package in NuGet Gallery. That package uses the Oracle Cloud Identity Data Plane Service (`OCI.DotNetSDK.Identitydataplane` NuGet package) and Oracle Cloud Identity (`OCI.DotNetSDK.Identity` NuGet package), which is automatically downloaded as a dependency.

To configure IAM SSO, ODP.NET requires IAM credential and connection information. It can use the authentication parameters in the table below to connect.

Table 3-3 ODP.NET OCI IAM Parameters

OCI IAM Parameter	Description
Token authentication	Specifies the authentication flow for retrieving an OCI IAM access token
Configuration file	Specifies the configuration file location for retrieving OCI IAM authentication values, such as user id and tenancy OCID
Profile	Specifies the profile ODP.NET uses from the configuration file
Compartment	Specifies the compartment Oracle Cloud ID for the database
Database	Specifies the database Oracle Cloud ID

For all OCI IAM authentication flows, only the token authentication parameter is required. All other parameters are either optional or not needed. The table below shows all the IAM authentication flows ODP.NET supports and the parameters available.

Table 3-4 ODP.NET OCI IAM Authentication Flows

OCI IAM Authentication Flows	Required Parameters
API Key	- Token authentication set to <code>OciApiKey</code> - Configuration file (optional) - Profile (optional) - Compartment OCID (optional) - Database OCID (optional)
Instance Principal	- Token authentication set to <code>OciInstancePrincipal</code> - Compartment OCID (optional) - Database OCID (optional)
Resource Principal	- Token authentication set to <code>OciResourcePrincipal</code> - Compartment OCID (optional) - Database OCID (optional)
Delegation Token	- Token authentication set to <code>OciDelegationToken</code> - Compartment OCID (optional) - Database OCID (optional)
Interactive	- Token authentication set to <code>OciInteractive</code> - Configuration File (optional) - Profile (optional) - Compartment OCID (optional) - Database OCID (optional) - Username (optional placeholder)
Default	Token authentication set to <code>OciDefault</code>

ODP.NET can retrieve the OCI IAM parameter values from any of the following configuration locations. The parameter values in the locations higher in the table have precedence over values in locations listed lower.

Table 3-5 ODP.NET OCI IAM Configuration Locations

Configuration Location	Available Parameters
OciTokenAuthentication class via OracleConnection	ConfigurationFile Profile Compartment Database OracleConnection.TokenAuthentication
Tnsnames.ora file or Easy Connect Plus	OCI_CONFIG_FILE OCI_PROFILE OCI_COMPARTMENT OCI_DATABASE TOKEN_AUTH

Table 3-5 (Cont.) ODP.NET OCI IAM Configuration Locations

Configuration Location	Available Parameters
OracleConfiguration class	OciConfigurationFile OciProfile OciCompartment OciDatabase TokenAuthentication
App/Web configuration file	OCI_CONFIG_FILE OCI_PROFILE OCI_COMPARTMENT OCI_DATABASE TOKEN_AUTH
Sqlnet.ora	OCI_CONFIG_FILE OCI_PROFILE OCI_COMPARTMENT OCI_DATABASE TOKEN_AUTH

ODP.NET OCI IAM SSO can be programmatically configured via the `OciTokenAuthentication` class, `OracleConnection UseOciTokenAuthentication` method, and `OracleConfiguration` class. The provider alters the `OciTokenAuthentication` object to read only after the connection is opened to ensure no OCI token authentication settings are changed for the pool's lifetime. With the settings locked, the token is cached for re-use when new pooled connections are requested.

ODP.NET OCI IAM SSO code sample:

```

OracleConnection conn = new OracleConnection(constr);
conn.TokenAuthentication = OracleTokenAuth.OciApiKey;
OciTokenAuthentication tokenconfig = new OciTokenAuthentication
{
    ConfigurationFile = "<CONFIGURATION FILE>",
    Profile = "<PROFILE>",
    Compartment = "<COMPARTMENT OCID>",
    Database = "<DATABASE OCID>"
}
conn.UseOciTokenAuthentication(tokenConfig);
conn.Open();
conn.Close();

```

Alternatively, ODP.NET OCI IAM SSO can be configured without any ODP.NET code changes using `tnsnames.ora`, `Easy Connect Plus`, `sqlnet.ora`, and/or `.NET` configuration files.

IAM Interactive Authentication Flows

For interactive authentication flows, when the user and tenancy name values are the same for two different data sources, the user will only need to login once even if the connections belong to different connection pools. The reason why is that the same token is used for all these connections and ODP.NET caches it. The app can direct ODP.NET to clear its cache of all its session tokens so that they aren't used with new connections or new pools. `ClearInteractiveTokenCache` static method will clear access token cached in existing

connection pools, that is, pools which are created with interactive authentication flows. Only existing opened connections will continue to work.

Code sample: ODP.NET OCI IAM Single Sign-on Connection Using Interactive Authentication Flow

```
string username = "user@company.com";
string constr = $"user id=<USERNAME>; data source = db-descriptor1";
OracleConnection conn = new OracleConnection(constr);
conn.TnsAdmin = @"C:\DB1\TnsAdmin\";
conn.TokenAuthentication = OracleTokenAuth.OciInteractive;
OciTokenAuthentication obj1 = new OciTokenAuthentication()
{
    ConfigurationFile = @"C:\oci_config\config"
};
conn.UseOciTokenAuthentication(obj1);
conn.Open();
Console.WriteLine("Connected to DB 1.");
conn.Close();

string constr2 = $"user id=<USERNAME>; data source = db-descriptor2 ";
OracleConnection conn2 = new OracleConnection(constr2);
conn2.TnsAdmin = @"C:\DB2\TnsAdmin\";

conn2.TokenAuthentication = OracleTokenAuth.OciInteractive;
OciTokenAuthentication obj2 = new OciTokenAuthentication()
{
    ConfigurationFile = @"C:\oci_config\config"
};
conn2.UseOciTokenAuthentication(obj2);
conn2.Open();
Console.WriteLine("Connected to DB 2.");
conn2.Close();
```

Using Microsoft Entra ID

Starting with ODP.NET 21.7 and 19.15.1, ODP.NET supports Microsoft Entra ID, formerly known as Azure Active Directory, authentication when connecting to Oracle Database. ODP.NET will then use an access token to access the database instead of a username and password.

This feature benefits applications and services that use Microsoft Entra ID for centralized user authentication with Oracle database. Those services can include Azure and Microsoft 365-based cloud services, such as Microsoft Power BI service, that rely on Microsoft Entra ID for user management.

Using token-based authentication is more secure and simpler for the end user. It becomes unnecessary to specify credentials each time the user accesses a resource. Moreover, the resource never needs to handle and manage individual user credentials.

Microsoft Entra ID follows the OAuth 2.0 user authorization standard. OAuth 2.0 provides a means of obtaining an access token, then using that access token for accessing resources, such as Oracle database.

Microsoft Entra ID is supported in the core, managed, and unmanaged ODP.NET drivers. It requires Oracle Autonomous Database or Oracle Database 19.16 or higher.

Managed and unmanaged ODP.NET require NET Framework 4.5.2 or higher for Microsoft Entra ID support. ODP.NET Core does not have a runtime requirement more specific than the general component system requirements to support Microsoft Entra ID.

Starting with ODP.NET version 23 and Oracle AI Database 26ai, Microsoft Entra ID single sign-on is supported with managed ODP.NET and ODP.NET Core. ODP.NET will automatically handle Microsoft Entra ID token retrieval and management.

Connecting Using Microsoft Entra ID Tokens

For an ODP.NET application using the managed or core provider to authenticate with Microsoft Entra ID using a file-based access token, the `TOKEN_AUTH` parameter or `OracleConfiguration.TokenAuthentication` property must be set to `OAuth`. The token's location is then specified in the `TOKEN_LOCATION` parameter or `TokenLocation` property. The token location must be specified explicitly when using `OAuth`. Otherwise, an error will occur. For `OAuth`, this location can be the directory where the file "token" is, or the full path that includes a file name.

For an ODP.NET application to authenticate with Microsoft Entra ID with an access token, none of these configuration steps is necessary. The access token is provided through ODP.NET APIs.

Developers construct an `OracleAccessToken` object to provide the access token via the `OracleConnection.AccessToken` property. If the application sets up a `RefreshAccessToken` event handler, ODP.NET will invoke it 60 seconds before the access token expires. The `OracleRefreshAccessTokenEventArgs` parameter provides a refreshed access token in the callback method.

The `OracleAccessToken` token can only be updated by calling `OpenWithNewToken` method or by setting the `OracleRefreshAccessTokenEventArgs.Token` property upon a .NET callback invocation.

For ODP.NET connection pooling, the `OracleAccessToken` object will be used to distinguish connection pools, not the actual token as they can expire and be refreshed. Only new connections need to use a refreshed access token. Existing pooled connections do not depend on the access token and will continue to work. Applications should always specify the same `OracleAccessToken` object on `OracleConnection` to ensure the same connection pool is used.

`TOKEN_AUTH` and `TOKEN_LOCATION` can also be set in the `tnsnames.ora` and `sqlnet.ora` files. To use `OAuth` with unmanaged ODP.NET, set your token values in these files as there are no equivalent unmanaged ODP.NET APIs.

Sample Code: Using ODP.NET Microsoft Entra ID Authentication

```
// This is a simple ODP.NET, Core Driver application that connects to an Oracle
Autonomous Database
// using a token obtained from Microsoft Entra ID.

// Azure.Identity can be obtained through NuGet Gallery.
// It will include the Azure.Core and Azure.Identity namespaces.
using System;
using System.Threading;
using Azure.Core;
using Azure.Identity;
using Oracle.ManagedDataAccess.Client;

namespace ConnectToOracleUsingAccessToken
{
    class Program
```

```

{
    static void Main()
    {
        try
        {
            // Retrieve an access token from Microsoft Entra ID.
            string token = GetAccessToken();

            // Create an instance of an OracleAccessToken. The access token needs to
            // be passed to the OracleAccessToken constructor as array of characters.
            var oracleAccessToken = new OracleAccessToken(token.ToCharArray());

            // Create an instance of an OracleConnection object.
            // The developer must provide the appropriate data source setting.
            var connection = new OracleConnection("User Id=;/;Data Source=<oracle>");

            // tnsnames.ora, sqlnet.ora, and cwallet.sso must reside in the same
            // directory as the application executable. These files can be downloaded
            // from Oracle Cloud for the Oracle Autonomous DB instance.
            connection.TnsAdmin = @".\";

            // Assign the OracleAccessToken to the AccessToken property on the
            // OracleConnection object.
            connection.AccessToken = oracleAccessToken;

            // Open the connection.
            connection.Open();

            // If Open() fails, it will throw an exception.
            Console.WriteLine("Open success.");

            // Dispose the OracleConnection object.
            connection.Dispose();
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex);
        }
    }
}

// Retrieves an Microsoft Entra ID access token through the
// Service Principal Auth flow using a client secret.
static string GetAccessToken()
{
    // The developer must configure the Microsoft Entra ID parameters below.
    string clientId = "<client Id of app registration in Microsoft Entra ID>";
    string tenantId = "<tenant Id of Microsoft Entra ID>";
    string clientSecret = "<secret value of app registration in Microsoft Entra ID>";
    string scope = "<scope of DB registration in Microsoft Entra ID>";

    // Create a TokenRequestContext object.
    var tokenRequestContext = new TokenRequestContext(new[] { scope });

    // Create a ClientSecretCredential object.
    var credentials = new ClientSecretCredential(tenantId, clientId, clientSecret);

    // Get the access token from Microsoft Entra ID.
    AccessToken accessToken = credentials.GetToken(tokenRequestContext,
    default(CancellationTokens));

    // Return the access token.
    return accessToken.Token;
}

```

```
}  
}  
}
```

See Also

More Microsoft Entra ID ODP.NET sample code can be found on Oracle .NET GitHub site:

<https://github.com/oracle/dotnet-db-samples/tree/master/samples/azure-active-directory>

Connecting Using Microsoft Entra ID Single Sign-On

ODP.NET supports Microsoft Entra ID single sign-on (SSO), allowing user identity to be propagated across Oracle databases and cloud services. SSO is easier for end users as they only need to log on once to access all the services that use an Microsoft Entra ID identity. It is also easier for .NET developers as ODP.NET performs access token management automatically.

When ODP.NET is given the Microsoft Entra ID authentication parameters, it acquires the Microsoft Entra ID access token to establish the database connection and all subsequent connection requests. The token is cached on a per connection pool basis. When the token is about to expire, ODP.NET automatically retrieves a new access token.

ODP.NET Microsoft Entra ID SSO requires `Oracle.ManagedDataAccess.Azure` package in NuGet Gallery. That package uses Azure .NET SDK (`Azure.Identity` NuGet package), which is automatically download as a dependency.

To configure Microsoft Entra ID SSO, ODP.NET requires Azure credential and connection information. It uses the parameters in the following table to connect.

Parameters	Description
Authentication method	This specifies the authentication flow for getting an Microsoft Entra ID bearer access token.
DB application ID URI	This specifies the protected resource identifier on Microsoft Entra ID for which client application requests the access token.
Tenant ID	This specifies tenant identifier under which applications (client app or database) are registered. The tenant is an entity or a logical group of cloud resources assigned to an individual or an organization.
Client ID	Client ID is the identifier of an application registered on Microsoft Entra ID.
Client secret	Client secret is the credential of a resource in Azure cloud. An app registered under Microsoft Entra ID can have a client secret.
Client certificate	Client certificate is the digital certificate of an Azure cloud resource. This also can be used instead of client secret.
Client certificate password	This specifies the protected client certificate password.

Parameters	Description
Redirect URI	A redirect URI, or reply URL, is the location where the authorization server sends the user once the app has been successfully authorized and granted an authorization code or access token. It is used in interactive authorization flow.
Azure username	This is the Azure accounts user's ID. For example: user@organization.onmicrosoft.com
Azure password	This is the Azure account's password.

The parameters required to connect depend on Azure authentication method and cloud account configuration. The following table shows all the Microsoft Entra ID authentication methods ODP.NET supports and the parameters each requires or can optionally use.

Authentication Methods	Required Parameters
Service Principal	- Client ID - Tenant ID - Client secret or client digital certificate path - Certificate password (only for protected certificates) - DB application ID URI
Interactive Authentication	- Client ID (optional when both client app and database registrations are under the same Azure tenant as the logged in user) - Tenant ID (optional when both client app and database registrations are under the same Azure tenant as the logged in user) - Redirect URI - DB application ID URI - Username (optional placeholder)
Managed Identity	- DB application ID URI - Client ID (only for user-defined managed identity)
Default	- DB application ID URI - Client ID [OPTIONAL] - Tenant ID [OPTIONAL] - Client secret [OPTIONAL] - Client digital certificate path [OPTIONAL] - Certificate password [OPTIONAL] - Azure account username [OPTIONAL] - Azure account password [OPTIONAL]
Username and Password	- Client ID - Tenant ID - DB application ID URI - Azure account username - Azure account password

Authentication Methods	Required Parameters
Device Code	- Client ID - Tenant ID - DB application ID URI

With Azure Default flow, the client driver evaluates five authentication flows in the sequence: Service Principal (using client secret), Service Principal (using client certificate), Username Auth Flow, Managed Identity, and Visual Studio Auth Flow. It performs the most appropriate flow based on the environment where the application is running.

Note

ODP.NET always uses the .default Microsoft Entra ID scope.

ODP.NET can attain these Microsoft Entra ID parameter values from any of the following configuration locations. The locations higher in the table have precedence over later locations listed lower.

Configuration Location	Available Parameters
AzureTokenAuthentication class	- ClientId - TenantId - DatabaseApplicationIdUri - ClientSecret - ClientCertificate - ClientCertificatePassword - RedirectUri
Tnsnames.ora file or Easy Connect Plus	- CLIENT_ID - TENANT_ID - AZURE_DB_APP_ID_URI - CLIENT_CERTIFICATE - REDIRECT_URI
OracleCredential class	Azure account username and password can be configured as username and password, respectively.
ODP.NET connection string	Azure account username and password can be configured as username and password, respectively.
SEPS wallet	Client ID and client secret can be configured as username and password, respectively. Azure account username and password can be configured as username and password, respectively.
Sqlnet.ora	- CLIENT_ID - TENANT_ID - AZURE_DB_APP_ID_URI

Configuration Location	Available Parameters
Azure .NET SDK environment variables	• AZURE_CLIENT_ID • AZURE_CLIENT_SECRET • AZURE_CLIENT_CERTIFICATE_PATH • AZURE_TENANT_ID • AZURE_USERNAME • AZURE_PASSWORD

ODP.NET Microsoft Entra ID SSO can be programmatically configured using the `OracleConnection UseAzureTokenAuthentication` extension method. The provider alters the `AzureTokenAuthentication` object to read only after the connection is opened to ensure no Azure token authentication settings are changed for the pool's lifetime. With the settings locked, the token is cached for re-use when new pooled connections are requested.

Azure Interactive Authentication Flows

For interactive authentication flows, when the user, application Id uri, client id, tenant id, and redirect uri values are same for two different connections, the user will only need to login once even if the connections belong to different connection pools. The reason why is that the same token is used for all these connections and ODP.NET caches it. The app can direct ODP.NET to clear its cache of all its access tokens so that they aren't used with new connections or new pools. `ClearInteractiveTokenCache` static method will clear access token cached in existing connection pools, that is, pools which are created with interactive authentication flows. Only existing opened connections will continue to work.

Code Sample: ODP.NET Microsoft Entra ID Single Sign-on Configuration and Connection

```

        string dbAppIdUri = "<value>";
        string clientId = "<value>";
        string tenantId = "<value>";
        string redirectUri = "<value>";
        var conn = new OracleConnection("User Id=/;Data Source=<value>;Connection
Timeout=900");
        var tokenConfig = new AzureTokenAuthentication
        {
            ClientId = clientId,
            TenantId = tenantId,
            DatabaseApplicationIdUri = dbAppIdUri,
            RedirectUri = redirectUri
        };
        conn.WalletLocation = @".\wallet";
        conn.TnsAdmin = @".\tns";
        conn.TokenAuthentication = OracleTokenAuth.AzureInteractive;
        conn.UseAzureTokenAuthentication(tokenConfig);
        conn.Open();

```

Alternatively, Microsoft Entra ID SSO can be configured without any ODP.NET code changes using `tnsnames.ora`, `Easy Connect Plus`, `sqlnet.ora`, `SEPS wallet`, and/or Azure .NET SDK environment variables.

Connection String Attributes and URLs

All ODP.NET providers support the connection string attributes in the following table.

Managed ODP.NET and ODP.NET Core can accept connection string URLs in lieu of connection string attributes. These URLs access the Azure, Oracle Cloud Infrastructure, or local file centralized configuration provider in which the ODP.NET connection string values are stored.

[Table 3-6](#) lists the supported connection string attributes.

Table 3-6 Supported Connection String Attributes

Connection String Attribute	Description	Default Value
Application Continuity	Enables database requests to automatically replay transactional or non-transactional operations in a non-disruptive and rapid manner in the event of a severed database session, which results in a recoverable error.	true
Connection Lifetime	Minimum life time (in seconds) of the connection.	0
Connection Timeout	The time to wait (in seconds) for a new connection or an idle connection from the connection pool before a connection time out error can occur.	15
Data Source	Oracle Net Services Name, Connect Descriptor, or an easy connect naming that identifies the database to which to connect.	empty string
DBA Privilege	Administrative privileges: SYSDBA, SYSASM, SYSOPER, SYSBACKUP, SYSDG, SYSKM, or SYSRAC.	empty string
Decr Pool Size	Number of connections that are closed when an excessive amount of established connections are unused.	1
Enlist	Controls the enlistment behavior and capabilities of a connection in context of COM+ transactions or System.Transactions.	true
HA Events	Enables ODP.NET connection pool to proactively remove connections from the pool when an Oracle database service, service member, instance, or node goes down. Works with Oracle Global Data Services, including Oracle RAC, Data Guard, GoldenGate, and some single instance deployments.	true
Load Balancing	Enables ODP.NET connection pool to balance work requests across Oracle database instances based on the load balancing advisory and service goal. Works with Oracle Global Data Services, including Oracle RAC, Active Data Guard, and GoldenGate.	true
Incr Pool Size	Number of new connections to be created when all connections in the pool are in use.	5
Max Pool Size	Maximum number of connections in a pool.	100
Metadata Pooling	Caches metadata information.	True
Min Pool Size	Minimum number of connections in a pool.	1

Table 3-6 (Cont.) Supported Connection String Attributes

Connection String Attribute	Description	Default Value
Password	Password for the user specified by User Id.	empty string
Persist Security Info	Retrieval of the password in the connection string.	false
Pooling	Connection pooling.	true
Proxy User Id	User name of the proxy user.	empty string
Proxy Password	Password of the proxy user.	empty string
Self Tuning	Enables or disables self-tuning for a connection.	true
Statement Cache Purge	Statement cache purged when the connection goes back to the pool.	false
Statement Cache Size	Statement cache enabled and cache size, that is, the maximum number of statements that can be cached.	0
Tns_Admin	Directory where ODP.NET can find its sqlnet.ora and tnsnames.ora configuration files. <i>Not available in ODP.NET, Unmanaged Driver.</i>	empty string
Token_Auth	This attribute specifies the access token authentication type. Possible values are OCITOKEN, OAUTH, or DISABLED. Only supported for managed ODP.NET and ODP.NET Core.	DISABLED
Token_Location	This attribute is the file-based token location. The value can be a directory where a file named "token" is or it can be the file's full path specification. Only supported for managed ODP.NET and ODP.NET Core.	Varies depending on token authentication type
User Id	Oracle user name.	empty string
Validate Connection	Validation of connections coming from the pool.	false
Wallet_Location	ODP.NET wallet directory or URL. <i>Not available in ODP.NET, Unmanaged Driver</i>	empty string
Wallet_Password	Stores PKCS #12 wallet password	empty string

The following example uses connection string attributes to connect to Oracle Database:

```
// C#

using System;
using Oracle.DataAccess.Client;

class ConnectionSample
{
    static void Main()
    {
        OracleConnection con = new OracleConnection();
```

```
//using connection string attributes to connect to Oracle Database
con.ConnectionString = "User Id=scott;Password=tiger;Data Source=oracle";
con.Open();
Console.WriteLine("Connected to Oracle" + con.ServerVersion);

// Close and Dispose OracleConnection object
con.Close();
con.Dispose();
Console.WriteLine("Disconnected");
}
}
```

📘 See Also

- [OracleConnection Properties](#) for detailed information on connection attributes
- [OracleCommand Object](#) for detailed information on statement caching
- [Centralized Configuration Providers for Deployments](#)

Connection String Builder

The `OracleConnectionStringBuilder` class makes creating connection strings less error-prone and easier to manage.

Using this class, developers can employ a configuration file to provide the connection string and/or dynamically set the values through the key/value pairs. One example of a configuration file entry follows:

```
<configuration>
  <connectionStrings>
    <add name="Publications" providerName="Oracle.DataAccess.Client"
        connectionString="User Id=scott;Password=tiger;Data Source=inst1" />
  </connectionStrings>
</configuration>
```

Connection string information can be retrieved by specifying the connection string name, in this example, `Publications`. Then, based on the `providerName`, the appropriate factory for that provider can be obtained. This makes managing and modifying the connection string easier. In addition, this provides better security against string injection into a connection string.

📘 See Also

[OracleConnectionStringBuilder Class](#)

Specifying the Data Source Attribute

This section describes different ways of specifying the data source attribute.

The following example shows a connect descriptor mapped to a TNS alias called `sales` in the `tnsnames.ora` file:

```
sales=
(DESCRIPTION=
  (ADDRESS= (PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))
  (CONNECT_DATA=
    (SERVICE_NAME=sales.us.acme.com)))
```

The connection pool will maintain the full descriptor of an alias so that subsequent connection requests with the same connection string will not need to resolve the alias again. This applies to `tnsnames.ora`, .NET config data sources, and LDAP aliases. To flush out the cached full descriptor maintained by the connection pool, invoke `OracleDataSourceEnumerator.GetDataSources()` followed by `OracleConnection.ClearPool()` or `OracleConnection.ClearAllPools()`.

If connection pooling is not used, the alias will need to be resolved to the full descriptor for each request. In the case of LDAP, the LDAP server is contacted for each connection request.

Using the TNS Alias

To connect as `scott/tiger` using the TNS Alias, a valid connection appears as follows:

```
"user id=scott;password=tiger;data source=sales";
```

Using the Connect Descriptor

ODP.NET also allows applications to connect without the use of the `tnsnames.ora` file. To do so, the entire connect descriptor can be used as the "data source".

The connection string appears as follows:

```
"user id=scott;password=tiger;data source=" +
  "(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)" +
  "(HOST=sales-server)(PORT=1521))(CONNECT_DATA="+
  "(SERVICE_NAME=sales.us.acme.com)))"
```

Easy Connect and Easy Connect Plus Naming Methods

The Easy Connect and Easy Connect Plus naming methods enable clients to connect to a database without any configuration.

With this enabled, ODP.NET allows applications to specify the Data Source attribute in the form of:

```
//host:[port]/[service_name]
```

Using the same example, some valid connection strings follow:

```
"user id=scott;password=tiger;data source=//sales-server:1521/sales.us.acme.com"
"user id=scott;password=tiger;data source=//sales-server/sales.us.acme.com"
"user id=scott;password=tiger;data source=sales-server/sales.us.acme.com"
```

If the port number is not specified, 1521 is used by default.

Easy Connect has been enhanced in ODP.NET 19c to support a wider application breadth, including clustered or cloud databases, and for ease of use. These improvements are called Easy Connect Plus and its features include:

- TCP/IP with SSL/TLS
- Any SQL*Net description level parameter can be used
- Multiple hosts and ports

- A straightforward name-value pair format

The syntax uses the question mark symbol ? to indicate the name-value pairs start and the ampersand symbol & to delimit each name-value pair. The entire connect string must be specified as a single string. Leading and trailing white spaces are ignored within parameter values. If whitespace is required as part of the value, it should be placed within double-quotes. In ODP.NET 23.5, Easy Connect Plus support for address lists was added.

Easy Connect Plus syntax:

```
[[protocol:][//]host1{,host2}[:port1]{,host3:port2}{;host4{,host5}[:port3]]
[/[service_name][:server][/instance_name]]{[?parameter_name=value{&parameter_name=value}]}
```

Easy Connect Plus samples:

- tcps://sales1.us.example.com:1522,sales2.us.example.com;sales3.us.example.com,sales4.us.example.com/MyServiceName?WALLET_LOCATION=/wallets
- tcps://salesserver1:1521/sales.us.example.com
- salesserver1:1521,salesserver2,salesserver3:1522/sales.us.example.com
- tcps://salesserver1:1521/sales.us.example.com?wallet_location=D:/oracle
- tcps://salesserver1:1521/sales.us.example.com?ssl_server_cert_dn=cn=sales,cn=OracleContext,dc=us,dc=example,dc=com
- tcps://salesserver1:1521/sales.us.example.com?https_proxy=www-proxy.mycompany.com&https_proxy_port=80
- salesserver1:1521/sales.us.example.com?connect_timeout=60&transport_connect_timeout=30&retry_count=3

① See Also

Oracle Database Net Services Administrator's Guide for details and requirements in the section Using Easy Connect Naming Method

Using LDAP

ODP.NET can connect with connect identifiers mapped to connect descriptors in an LDAP-compliant directory server, such as Oracle Internet Directory and Microsoft Active Directory.

To enable ODP.NET LDAP connectivity, provide configuration information using an `ldap.ora` file.

To configure LDAP for ODP.NET, Unmanaged Driver, follow these Oracle documentation instructions in Configuring the Directory Naming Method in *Oracle AI Database Net Services Administrator's Guide*.

To configure LDAP for ODP.NET, Managed Driver, follow the instructions in "[settings section](#)" and "[LDAPsettings section](#)."

Beginning with Oracle Database release 18c, version 18.1, organizations can use centrally managed users (CMUs) with Active Directory. This feature is designed for organizations who prefer to use Active Directory as their centralized identity management solution. Organizations can use Kerberos, PKI, or password authentication with CMU and Active Directory.

LDAP channel binding and LDAP signing are ways to enhance the communication security between LDAP clients and Active Directory domain controllers. Beginning with version 19,

managed ODP.NET and ODP.NET Core LDAP Naming adapters fully comply with the Microsoft LDAP hardening guidance.

① See Also

- *Oracle AI Database Net Services Administrator's Guide* and *Oracle AI Database Security Guide* for details and requirements in the section Using LDAP
- Microsoft Guidance for [Enabling LDAP Channel Binding and LDAP Signing](#)

LDAP URL Naming

LDAP URL Naming enable clients to resolve a TNS alias stored in a LDAP server without SQLNET.ORA and LDAP.ORA configuration files.

Supported Syntax:

```
ldap[s]://host[:port]/name[,context][?parameter=value{&parameter=value}]
```

host is host name or IP address.

port is port used. Default port number is 636 for LDAPS, 363 for LDAP.

name[,context] can be:

- alias, Default_Admin_Context
- alias, CN=OracleContext, Default_Admin_Context
- CN=alias, Default_Admin_Context
- CN=alias, CN=OracleContext, Default_Admin_Context

Default_Admin_Context is the value of the DEFAULT_ADMIN_CONTEXT parameter from LDAP.ORA.

parameters:

- DIRECTORY_SERVER_TYPE can be OID, OUD, AD, with OID as the default.
- AUTHENTICATE_BIND can be true or false, with false as the default.
- AUTHENTICATE_BIND_METHOD can be LDAPS_SIMPLE_AUTH or NONE, to indicate to use user name/password from the wallet.
- WALLET_LOCATION specifies the location of the wallet. If WALLET_LOCATION is missing in the URL and required, it will be retrieved from the SQLNET.ORA.
- LDAP_CONN_TIMEOUT specifies the LDAP connection timeout in seconds.

When LDAP URL is used to resolve an alias, all LDAP related parameters in SQLNET.ORA and LDAP.ORA are ignored with the exception of WALLET_LOCATION parameter in SQLNET.ORA.

LDAPS Configuration without Wallets

Starting with versions 23.4, LDAP Naming adapter for ODP.NET managed and core drivers supports SSL without wallets. With one-way TLS, the LDAP client certificate is not required. A client wallet for the trusted Certificate Authority (CA) can be provided for ODP.NET LDAP Naming adapter authentication of the server. Without a wallet, the trusted CA will come from the local host's default trustpoints. LDAPS connection configuration is simpler without wallet files.

LDAPS Walletless One-way TLS can be used in one of two ways:

- If the LDAP server uses a certificate/wallet signed by a O/S-trusted certificate authority, as defined by the .NET root/localmachine store, then for LDAP client no changes are required besides un-setting the Oracle wallet location if it is already set.
- If the LDAP server uses a self-signed certificate/wallet, then the self-signed root certificate (public key) for the LDAP server certificate/wallet must be inserted into the root certificate store (root/localmachine) of the client machine.

ODP.NET LDAP Naming adapter will attempt to connect with one-way TLS when TCPS protocol is specified and either:

- no wallet location setting is present, or,
- wallet location is set to system.

Data Source Enumerator

The data source enumerator enables the application to generically obtain a collection of the Oracle data sources that the application can connect to.

See Also

["OracleDataSourceEnumerator Class"](#)

Using WebSocket

Websocket is a protocol that offers full-duplex communication channels over a single TCP connection. WebSocket with SSL/TLS offers a secure WebSocket connection. WebSocket is an extension to HTTP and is able to work with HTTP proxies and intermediaries.

ODP.NET Core, managed, and unmanaged providers all support Websocket and secure WebSocket protocols in Oracle Database 19c and higher.

ODP.NET WebSocket Configuration

WebSocket and secure WebSocket can be set through the connect descriptor by setting PROTOCOL to WS for WebSocket or WSS for secure WebSocket in tnsnames.ora, .NET configuration file, or OracleDataSourceCollection Class.

The WebSocket uniform resource identifier (URI) can be set in tnsnames.ora, sqlnet.ora, .NET configuration file, and OracleConfiguration class.

ODP.NET Configuration File WebSocket URI Setting Sample:

```
<oracle.manageddataaccess.client>
  <version number="">
    <settings>
      <setting name="SQLNET.URI" value="<WebSocket URI>" />
    </settings>
  </version>
</oracle.manageddataaccess.client>
```

OracleConfiguration WebSocket URI Setting C# Sample:

```
OracleConfiguration.SqlNetURI = <WebSocket URI>;
```

When setting the URI in unmanaged ODP.NET, it cannot begin with a forward slash character. For managed ODP.NET, the forward slash at the beginning of the URI is optional.

Using Transport Layer Security and Secure Sockets Layer

Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are industry standard protocols for securing network connections.

ODP.NET core, managed, and unmanaged versions support one-way and two-way TLS/SSL with wallets for database and transport authentication. ODP.NET core, managed, and unmanaged also support one-way TLS without wallets. Not having to provide a wallet can simplify database connectivity, such as with Oracle Autonomous Database.

Secure Sockets Layer and Transport Layer Security Differences

Although SSL was primarily developed by Netscape Communications Corporation, the Internet Engineering Task Force (IETF) took over development of it, and renamed it Transport Layer Security (TLS).

ODP.NET supports TLS 1.2 and 1.3.

The SSL/TLS client can ensure that the distinguished name (DN) is correct for the database server it is trying to connect to. Parameters for DN Matching are `SSL_SERVER_DN_MATCH` (`sqlnet.ora`) and `SSL_SERVER_CERT_DN` (`tnsnames.ora`), which can be defined in the `.NET` config file as well.

To turn DN Match on, set `SSL_SERVER_DN_MATCH` to True (or On or Yes). `SSL_SERVER_CERT_DN` is optional. It allows the administrator to specify exactly the DN they want to match. If the `SSL_SERVER_CERT_DN` is not set, then `HOSTNAME` based DN matching will be done.

`SSL_SERVER_DN_MATCH` can be set using:

- `sqlnet.ora`
- connect address: *Both* full description based and EZConnect
- `.NET` Configuration file, such as `web.config`
- `SSLServerDNMatch` in `OracleConfiguration` and `OracleConnection`
- `UseSSLServerDNMatch` in `OracleDataSourceBuilder`

`SSL_SERVER_CERT_DN` can be set using:

- connect address: *Both* full description based and EZConnect

See Also

- The TLS Protocol Version 1.0 [RFC 2246] at the IETF Web site, which can be found at:
<http://www.ietf.org>
- `SSL_VERSION` in the "[settings section](#)."

Note

To simplify the discussion, this section uses the term SSL where either SSL or TLS may be appropriate because SSL is the most widely recognized term. However, where distinctions occur between how you use or configure these protocols, this section specifies what is appropriate for either SSL or TLS.

ODP.NET Secure Sockets Layer Configuration Using Wallets

When you configure Secure Sockets Layer on the client, you must confirm that the wallet is created and use TCP/IP with SSL on the client. Optionally, you can perform additional steps to enhance the configuration.

Note

ODP.NET supports auto login wallets, but not the local auto login wallet option.

SSL Configuration Topics:

- [Step 1: Confirm Client Wallet Creation](#)
- [Step 2: Use TCP/IP with SSL on the Client](#)
- [Step 3: Specify Required Client SSL Configuration \(Wallet Location\)](#)
- [Step 4: Set the SSL Version on the Client \(Optional\)](#)
- [Step 5: Set SSL as an Authentication Service on the Client \(Optional\)](#)

Step 1: Confirm Client Wallet Creation

Before proceeding to the next step, you must confirm that a wallet has been created on the client and that the client has a valid certificate.

ODP.NET, managed and core drivers support file and Microsoft Certificate Store (MCS) based wallets.

- For file-based wallets, use Oracle Wallet Manager to check that the wallet has been created. See Step 1A: Confirm Wallet Creation on the Server in *Oracle Database Security Guide* for information about checking a wallet.
- For MCS, ODP.NET will retrieve the credentials from the MY or Personal certificate store. Use Microsoft tools or the `orapki` utility to create certificates, then load the certificates into MCS for use.

Step 2: Use TCP/IP with SSL on the Client

The ODP.NET Data Source must be modified to use SSL. Specifically, the transport protocol must be changed to use TCP/IP with SSL or what Oracle calls "tcps". An example ODP.NET Data Source for use with SSL is:

```
finance = (DESCRIPTION=
  (ADDRESS = (PROTOCOL=tcps) (HOST=finance_server) (PORT=1575) )
  (CONNECT_DATA = (SERVICE_NAME=Finance.us.example.com) ) )
```


Step 3: Specify Required Client SSL Configuration (Wallet Location)

Edit the `sqlnet.ora` or `.NET` application configuration to specify the wallet location.

- An example of setting the SSL wallet location for file based wallets, where `<wallet_location>` is the specified location where the client wallet is stored:

```
wallet_location = (SOURCE=(METHOD= File)
                  (METHOD_DATA=(DIRECTORY=<wallet_location>)))
```

- An example of setting the SSL wallet location for MCS based wallets is:

```
wallet_location = (SOURCE=(METHOD= MCS))
```

Step 4: Set the SSL Version on the Client (Optional)

The `SSL_VERSION` parameter can be set through the `sqlnet.ora` or the `.NET` `application.config`, `web.config`, or `machine.config` file. Normally, it is not necessary to set this parameter. The default setting for this parameter is `any`, which allows the database server to apply any necessary restrictions to the SSL version accepted. An example setting in the `sqlnet.ora` is:

```
SSL_VERSION=1.2
```

Step 5: Set SSL as an Authentication Service on the Client (Optional)

If TCPS is to be used as both a transport and as an external database authentication, set the `SQLNET.AUTHENTICATION_SERVICES` parameter in the `sqlnet.ora` or `application.config`, `web.config`, or `machine.config` file.

Note that SSL can be used as just a transport encryption vehicle. Hence, the "optional" designation for this setting.

If SSL/TLS is to be used as an external database authentication method, then an externally authenticated database user matching the client certificate must be created.

An example setting allowing SSL external authentication in the `sqlnet.ora` is:

```
SQLNET.AUTHENTICATION_SERVICES = (TCPS)
```

① Note

Prior to ODAC 12c Release 4, ODP.NET, Managed Driver SSL connections would be redirected to dynamic (ephemeral) port on the database server machine. With ODAC 12c Release 4 and later, managed ODP.NET SSL connections will now continue to the original socket connection to the Oracle Listener. Hence, firewalls will now only need to allow access to the Oracle Listener's port (e.g., 1521).

① See Also

- Enabling Secure Sockets Layer in *Oracle Database Security Guide* for more information about TLS/SSL and configuration with Oracle database.
- Creating a User Who Is Authenticated Externally in *Oracle Database Security Guide* for more information about creating externally identified database users.

ODP.NET Secure Sockets Layer Configuration without Wallets

Starting with versions 19.13 and 21.4, ODP.NET managed and core drivers support SSL without wallets. Unmanaged ODP.NET introduced SSL support without wallets from versions 19.14 and 21.5. With one-way TLS, the client certificate is not required. A client wallet for the trusted Certificate Authority (CA) can be provided for ODP.NET authentication of the server. Without a wallet, the trusted CA will come from the local host's default trustpoints. Connection configuration is simpler without wallet files.

Walletless One-way TLS can be used in one of two ways:

- If the database server uses a certificate/wallet signed by a O/S-trusted certificate authority, as defined by the .NET root/localmachine store, then no client changes are required besides un-setting the Oracle wallet location if it is already set.
- If the database server uses a self-signed certificate/wallet, then the self-signed root certificate (public key) for the server certificate/wallet must be inserted into the root certificate store (root/localmachine) of the client machine.

ODP.NET will attempt to connect with one-way TLS when TCPS protocol is specified and either:

- no wallet location setting is present, or,
- wallet location is set to system.

ODP.NET essentially ignores existing static wallet configuration (file or `OracleConfiguration` based) when system is specified through the following dynamic configuration methods:

- TNS connect descriptor
 - `WALLET_LOCATION` or `MY_WALLET_DIRECTORY`
- Easy Connect Plus
 - `WALLET_LOCATION`
- ODP.NET connection string
 - `Wallet_Location`
- ODP.NET `OracleConnection` property
 - `WalletLocation`

These options allow administrators to specify at different granularities connections using a client certificate or SEPS and connections not using a wallet.

By default, Oracle Autonomous Database credentials use the `WALLET_LOCATION` in `sqlnet.ora`. To disable the wallet requirement, remove the `sqlnet.ora` `WALLET_LOCATION` setting completely or set one of the above dynamic wallet location settings to system.

Inserting Public Keys into System Trusted Certificate Authority List

The Certificate Authority (CA) public key is stored in the CA certificate. This certificate is stored in the local trust store on the client system. Here are typical commands to install root certificate in different platform.

Windows:

Base64/PEM format:

```
openssl pkcs12 -in ewallet.p12 -clcerts -nokeys -out rootca.pem  
certutil -addstore -enterprise -f -v root rootca.pem
```

Linux (Oracle and Red Hat):

Create PEM certificate from PKCS #12 file:

```
openssl pkcs12 -in ewallet.p12 -clcerts -nokeys -out rootca.pem -password pass:<password>  
sudo cp rootca.pem /etc/pki/ca-trust/source/anchors  
sudo update-ca-trust
```

On Linux, the administrator can manually edit the root PEM file by adding the PEM file contents resulting from the OPENSSL command shown above. The root PEM file on Linux is located in /etc/pki/tls/cert.pem, which is a link to /etc/pki/ca-trust/extracted/pem/tls-ca-bundle.pem.

Linux (Ubuntu):

Change the PEM (.pem) or CER (.cer) file extension to a CRT extension (.crt). Then, execute the following two commands:

```
sudo cp rootca.crt /usr/local/share/ca-certificates/rootca.crt  
sudo update-ca-certificates
```

macOS:

Base64/PEM format:

```
sudo security add-trusted-cert -d -r trustRoot -k "/Library/Keychains/System.keychain"  
"<directory>/rootca.pem"
```

Troubleshooting TLS/SSL Setup

This section discusses commonly encountered issues and their typical resolution steps.

Common TLS/SSL Wallet Errors

Microsoft Windows now restricts wallets from using the MD5 algorithm. Oracle wallets may have been generated with this algorithm as that was the default option in Oracle Public Key Infrastructure (orapki) utility 12.1 and earlier.

orapki can be found in the ORACLE_HOME\bin directory of the database server installation or Oracle client administrator install. It is not included with Oracle Instant Client. The utility is only needed to setup up the wallet; it is not necessary to deploy it with the wallet.

When you setup TLS/SSL and encounter an "ORA-0052: Failure during SSL handshake" error combined with a 0x80004005 error code and first inner exception "A SSPI-call failed" and second inner exception "A token sent to the function is invalid", then it is very likely that Microsoft Security Support Provider Interface (SSPI) rejected your Oracle Wallet, such as when MD5 is used. This is a failure on the handshake. You can resolve this error by using the SHA-2 algorithm instead.

If the second inner exception instead indicates "The credentials supplied to the package were not recognized", it is possible the user certificate was generated without a certificate authority (CA). You can resolve this error by using orapki to generate a CA/root certificate and then regenerating your user wallet/certificate to point to this new CA/root certificate.

The steps below will regenerate your Oracle Wallet using orapki and SHA-2. Any orapki version can be used to generate the wallet with these instructions.

1. Create root wallet, for example, a CA wallet.

```
orapki wallet create -wallet ./root -pwd <password>
```

2. Add a self-signed certificate (CA certificate) to the root wallet.

```
orapki wallet add -wallet ./root -dn 'CN=<my root>' -keysize 1024 -  
self_signed -validity 3650 -pwd <password> -sign_alg sha512
```

3. Export the self-signed certificate from the wallet.

```
orapki wallet export -wallet ./root -dn 'CN=<my root>' -cert ./root/  
b64certificate.txt -pwd <password>
```

4. Create a user wallet, for example, a customer wallet.

```
orapki wallet create -wallet ./user -pwd <password> -auto_login
```

5. Add a certificate request.

```
orapki wallet add -wallet ./user -dn 'CN=<client's hostname>' -keysize  
1024 -pwd <password> -sign_alg sha512
```

6. Export the certificate request.

```
orapki wallet export -wallet ./user -dn 'CN=<client's hostname>' -  
request ./user/creq.txt -pwd <password>
```

7. Create a certificate issued by a CA.

```
orapki cert create -wallet ./root -request ./user/creq.txt -cert ./user/  
cert.txt -validity 3650 -pwd <password> -sign_alg sha512
```

8. Add a trusted certificate (CA certificate) to the wallet. This example assumes the same CA for both the client and server wallets.

```
orapki wallet add -wallet ./user -trusted_cert -cert ./root/  
b64certificate.txt -pwd <password>
```

9. Add a user certificate.

```
orapki wallet add -wallet ./user -user_cert -cert ./user/cert.txt -pwd  
<password> -sign_alg sha512
```

10. Display contents of user wallet.

```
orapki wallet display -wallet ./user -pwd <password>
```

11. Create a server wallet.

```
orapki wallet create -wallet ./server -pwd <password> -auto_login
```

12. Add a server certificate request.

```
orapki wallet add -wallet ./server -dn 'CN=<server's hostname>' -keysize  
1024 -pwd <password> -sign_alg sha512
```

13. Export the certificate request.

```
orapki wallet export -wallet ./server -dn 'CN=<server's hostname>' -  
request ./server/creq.txt -pwd <password>
```

14. Create a server certificate issued by a CA.

```
orapki cert create -wallet ./root -request ./server/creq.txt -cert ./  
server/cert.txt -validity 3650 -pwd <password> -sign_alg sha512
```

15. Add a trusted certificate (CA certificate) to the server wallet. This example assumes the same CA for both the client and server wallets.

```
orapki wallet add -wallet ./server -trusted_cert -cert ./root/  
b64certificate.txt -pwd <password>
```

16. Add an user_cert certificate for the server wallet.

```
orapki wallet add -wallet ./server -user_cert -cert ./server/cert.txt -pwd  
<password> -sign_alg sha512
```

17. Display contents of server wallet.

```
orapki wallet display -wallet ./server -pwd <password>
```

Mutual TLS/SSL Certificate Selection

When ODP.NET connections use mutual TCPS (TLS/SSL), the user needs a signed certificate. This client certificate can be stored in an Oracle wallet, PEM file, or Microsoft Certificate Store (MCS).

PEM files are supported by ODP.NET Core only starting with version 23.9.

In MCS, if there is more than one certificate that can be used, then the user or application settings must specify the specific one to connect with. This choice can be made manually by the user through the graphical user interface (GUI) or automatically by the application using the thumbprint.

For the GUI, the end user selects the correct client certificate from the list available in a store, such as MCS. The key store can filter the visible certificates to simplify selecting the correct certificate. For example, the GUI should be able to filter certificates by private certificate and expiration state. By default, public key only trusted Certificate Authorities (CA) certificates should not be visible in the certificate store GUI as the selection process is for mutual TLS.

GUI certificate selection is disabled by default. To enable, set the `OracleConnection.AllowCertificateSelectionUI` property to true. When connecting from Windows, the end user gets a pop-up window to select a TLS/SSL certificate in MCS.

Alternatively to GUI selection, a thumbprint can uniquely identify the client certificate.

Developers can set the thumbprint in the `SSLCertificateThumbprint` property of `OracleConfiguration` or `OracleConnection`, TNS connect descriptor, Easy Connect Plus, or `sqlnet.ora` entries.

`orapki` (`orapki.exe`) can manage certificates and wallets for Oracle database. It can create, alter, view certificates in the Oracle wallet to include the SHA1 thumbprints. The thumbprint of a wallet can be displayed via the following `orapki` command:

```
orapki wallet display -wallet <wallet directory> -pwd <wallet password>
      -complete
```

Microsoft Windows automatically adds Personal Identity Verification (PIV) smart card certificates into the Windows MCS. When those smart card certificates are used, such as during a TLS/SSL connection, they require a successful PIN prompt entry. The initial PIN prompt success is cached at the process level for all subsequent same smart card certificate uses, except for "digital signature" PIV smart card certificate containers. For certificates in these containers, a PIN prompt is required for every certificate use.

① See Also

[Registry keys for smart card PIN caching options are no longer available in Windows 10](#)

Using Secure External Password Store

The Secure External Password Store (SEPS) is the use of a client-side wallet for securely storing the password credentials. All ODP.NET driver types can be configured to use the external password store.

An Oracle wallet is a container that securely stores authentication and signing credentials. Wallets can simplify large-scale deployments that rely on password credentials for database connections. Applications no longer need embedded user names and passwords, which reduces security risk.

Using SEPS with ODP.NET Core requires the NuGet package, `System.Security.Cryptography.Pkcs` version 4.7 or higher, as a dependency for your application project.

ODP.NET can use SEPS in conjunction with TCPS in two scenarios:

- SEPS wallet from `sqlnet.ora` `WALLET_LOCATION` and TCPS wallet from connect address `WALLET_LOCATION`
- Only SEPS wallet from `sqlnet.ora` `WALLET_LOCATION` is specified and `WALLET_LOCATION` is *not* specified in connect address

In the second scenario, if the wallet has a client certificate, mTLS is used. If there is no certificate, then walletless one-way TLS is used.

SEPS typically associates a TNS alias with a database username and password encrypted in the wallet file. When the TNS alias-based SEPS username and password are not found, a SEPS default username and password can be configured in the wallet as a backup option. ODP.NET would then use the default username and password instead to connect. ODP.NET support for default username and password requires version 23.8 or higher.

Also, starting in version 23.8, ODP.NET managed and core drivers support SEPS wallet location and password. These capabilities allow apps to separate the SEPS wallet

configuration from the database, LDAP, and WebSocket Secure wallet support. This decouples the SEPS configuration, allowing it to change independently from other wallet uses. SEPS wallet password is only required if you use a PKCS #12 wallet file, which typically has .p12 extension.

Configuring Secure External Password Store (SEPS)

Steps for configuring SEPS:

- [Step 1. Create the wallet file](#)
- [Step 2. Point the configuration to the client wallet](#)
- [Step 3. Turn on SEPS](#)

Step 1. Create the wallet file

Use the `mkstore` utility to create the wallet file and insert the credentials.

Step 1a. Create a wallet on the client by using the following syntax at the command line:

```
mkstore -wrl wallet_location -create
```

For example:

```
mkstore -wrl c:\oracle\product\12.1.0\db_1\wallets -create
Enter password: password
```

Step 1b. Create database connection credentials in the wallet by using the following syntax at the command line:

```
mkstore -wrl wallet_location -createCredential db_connect_string username
Enter password: password
```

For example:

```
mkstore -wrl c:\oracle\product\12.1.0\db_1\wallets -createCredential orcl system
Enter password: password
```

Step 2. Point the configuration to the client wallet

In the client `sqlnet.ora` file, enter the `WALLET_LOCATION` parameter and set it to the directory location of the wallet you created in Step 1.

For example, if you created the wallet in `$ORACLE_HOME/network/admin` and your Oracle home is set to `C:\app\client\<user>\product\<version>\client_1\`, then you need to enter the following into your client `sqlnet.ora` file:

```
WALLET_LOCATION =
  (SOURCE =(METHOD = FILE)
    (METHOD_DATA =
      (DIRECTORY =
        C:\app\client\<user>\product\<version>\client_1\Network\Admin) ) )
```

Step 3. Turn on SEPS

Enable SEPS in your application by setting the wallet override setting to `true` in the `OracleConfiguration.SqlNetWalletOverride` property or `SQLNET.WALLET_OVERRIDE` setting in `sqlnet.ora` or `.NET` configuration file.

```
SQLNET.WALLET_OVERRIDE = TRUE
```

This setting causes all `CONNECT /@db_connect_string` statements to use the information in the wallet at the specified location to authenticate to databases.

When external authentication is in use, an authenticated user with such a wallet can use the `CONNECT /@db_connect_string` syntax to access the previously specified databases without providing a user name and password. Note however, that the wallet file needs to be kept up to date with the database credentials. If the database credentials change, but the wallet file is not changed appropriately, then the connections will fail.

See Also

- [Oracle Data Provider for .NET, Managed Driver Configuration](#) for more details about the wallet settings.
- Managing the Secure External Password Store for Password Credentials in *Oracle Database Security Guide* for more information about secure external password store and configuration with Oracle database.

Using Kerberos

Kerberos is a network authentication service for security in distributed environments. ODP.NET applications can use Kerberos for single sign-on and centralized user authentication.

All ODP.NET provider types support Kerberos for external authentication to the database server.

ODP.NET Core Kerberos requires the `Oracle.ManagedDataAccess.Kerberos` NuGet package, which uses `Kerberos.NET`. When using ODP.NET Core Kerberos, ensure that the local hostname is IP resolvable via DNS or a local hosts file.

Managed ODP.NET offers two Kerberos enablement options. Developers can use the `Oracle.ManagedDataAccess.Kerberos` NuGet package with `Kerberos.NET` just like they do for ODP.NET Core or they can use MIT Kerberos. Oracle recommends using the `Oracle.ManagedDataAccess.Kerberos` option.

Note

- Managed ODP.NET and ODP.NET Core do not support Kerberos constrained delegation.
- Managed ODP.NET and ODP.NET Core do not support Kerberos5Pre authentication adapter. Use Kerberos5 instead.

File Based Credential Cache and MSLSA

ODP.NET supports both a file-based Kerberos client credential cache (CC) and the ability to use Windows logon credentials as Kerberos client credentials. The latter is called MSLSA-based Kerberos authentication.

In order to utilize a file based Kerberos client credential cache (CC), the following executables associated with the full Oracle Database Client install are needed:

- `okinit.exe`

- `oklist.exe`
- `okdstry.exe`

The executables are required in order to acquire the Kerberos5 credentials and store them in the file based credential cache (CC). However, after credential cache creation, as long as the credentials remain valid, the above executables are then unneeded by the ODP.NET application at run-time.

Using ODP.NET, Managed Driver with Kerberos

Managed ODP.NET supports Kerberos authentication with either MIT Kerberos (default) or `Oracle.ManagedDataAccess.Kerberos` NuGet package. Managed ODP.NET 23.9 and higher support the `Oracle.ManagedDataAccess.Kerberos` option.

To use `Oracle.ManagedDataAccess.Kerberos`, set `OracleConfiguration.UseMITKerberos` to false and add the `Oracle.ManagedDataAccess.Kerberos` NuGet package into your project.

If set to true (default), then managed ODP.NET will use MIT Kerberos instead. You then install MIT Kerberos 4.0.1 onto the ODP.NET client machine. The software can be downloaded from the [MIT Website](#).

① See Also

Configuring Kerberos Authentication in *Oracle Database Security Guide* for more information about Kerberos and configuration with Oracle database.

Configuring Kerberos Authentication with ODP.NET

Please reference the following "key" when viewing the below Kerberos configuration examples:

- `oracleclient` = Kerberos/Windows Domain user ID used by the Oracle database client program to represent the Oracle Client user on the domain
- `oracleserver` = Kerberos/Windows Domain user ID used by the Oracle database server
- `DOMAIN.COMPANY.COM` = Kerberos/Windows domain
- `dbhost.company.com` = Oracle database server machine hostname
- `kerberos_service_name` = Kerberos service name
- `dc.company.com` = hostname for Kerberos Key Distribution Center (KDC) and Windows Domain Controller

Configuring Kerberos Authentication Topics:

- [Step 1. Update Windows services file to include a "kerberos5" entry](#)
- [Step 2. Create client and server Kerberos users \(Windows domain users for MSLSA\)](#)
- [Step 3. Associate the DB server's Kerberos principal name with the DB server's Kerberos Service \(SPN mapping\) and generate the server keytab file](#)
- [Step 4. Confirm the mapping of server user to service principal](#)
- [Step 5. Setup server `sqlnet.ora` to point to the keytab file generated in step 2](#)

- [Step 6. Create a kerberos configuration file that points to the Kerberos KDC \(Windows Domain Controller for MSLSA\)](#)
- [Step 7. Configure the Oracle database client and server sqlnet.ora or .NET config to point to the above Kerberos configuration file](#)
- [Step 8. Point the client sqlnet.ora or .NET config to a credential cache file or to MSLSA](#)
- [Step 9. Set the client and server authentication services in the sqlnet.ora or .NET config to Kerberos5](#)
- [Step 10. Setup an externally authenticated database user that matches the Kerberos client user setup in step 1 \(note the case\)](#)
- [Step 11. Login to the client machine via the Windows Domain client user \(for MSLSA\) or perform an okinit to authenticate the client Kerberos user \(for file based CC\):](#)
- [Step 12. Run the ODP.NET application](#)

Step 1. Update Windows services file to include a "kerberos5" entry

Change the Kerberos entry in the Windows service file
(C:\windows\system32\drivers\etc\services) from:

```
kerberos    88/tcp          krb5 kerberos-sec      #Kerberos
```

to:

```
kerberos    88/tcp kerberos5 krb5 kerberos-sec      #Kerberos
```

Step 2. Create client and server Kerberos users (Windows domain users for MSLSA)

As noted in the above "key", we will use oracleclient and oracleserver as our client and server Kerberos user IDs, respectively.

ODP.NET supports MSLSA using Windows domain users which have the following attributes:

- "Kerberos DES" unchecked
- "Kerberos AES 128 bit" checked
- "Kerberos AES 256 bit" checked
- "Kerberos preauthentication not required" checked

Step 3. Associate the DB server's Kerberos principal name with the DB server's Kerberos Service (SPN mapping) and generate the server keytab file

Run the following commands on the Kerberos KDC (Windows Domain Controller for MSLSA) as an administrator:

```
> ktpass -princ kerberos_service_name/dbhost.company.com@DOMAIN.COMPANY.COM /crypto all /mapuser oracleserver@DOMAIN.COMPANY.COM /pass <oracleserver password> /out v5srvtab
```

```
> setspn -A kerberos_service_name/dbhost.company.com@DOMAIN.COMPANY.COM oracleserver
```

Step 4. Confirm the mapping of server user to service principal

Also on the Kerberos KDC, run the following command, noting the output:

```
> setspn -L oracleserver
```

```
Registered ServicePrincipalNames for  
CN=oracleserver,CN=Users,DC=domain,DC=company,DC=com:
```

```
kerberos_service_name/dbhost.company.com
kerberos_service_name/dbhost.company.com@DOMAIN.COMPANY.COM
```

Step 5. Setup server sqlnet.ora to point to the keytab file generated in step 2

Add the following line to the server sqlnet.ora:

```
sqlnet.kerberos5_keytab = c:\krb\kv5srvtab
```

Step 6. Create a kerberos configuration file that points to the Kerberos KDC (Windows Domain Controller for MSLSA)

An example kerberos configuration file (krb.conf):

```
[libdefaults]
default_realm = DOMAIN.COMPANY.COM

[realms]
DOMAIN.COMPANY.COM = {
    kdc = dc.company.com
}

[domain_realm]
.domain.company.com = DOMAIN.COMPANY.COM
domain.company.com = DOMAIN.COMPANY.COM
.DOMAIN.COMPANY.COM = DOMAIN.COMPANY.COM
DOMAIN.COMPANY.COM = DOMAIN.COMPANY.COM
```

Step 7. Configure the Oracle database client and server sqlnet.ora or .NET config to point to the above Kerberos configuration file

Edit the client or server sqlnet.ora to include:

```
sqlnet.kerberos5_conf = C:\krb\krb.conf
```

Or edit the client application config to include (in the [settings section](#)):

```
<setting name="sqlnet.kerberos5_conf" value="C:\krb\krb.conf" />
```

Step 8. Point the client sqlnet.ora or .NET config to a credential cache file or to MSLSA

Example pointing to Credential Cache file:

```
sqlnet.kerberos5_cc_name = c:\krb\krb.cc
```

Example pointing to MSLSA:

```
sqlnet.kerberos5_cc_name = MSLSA:
```

Step 9. Set the client and server authentication services in the sqlnet.ora or .NET config to Kerberos5

```
sqlnet.authentication_services=(Kerberos5)
```

Step 10. Setup an externally authenticated database user that matches the Kerberos client user setup in step 1 (note the case)

```
create user "ORACLECLIENT@DOMAIN.COMPANY.COM" identified externally;
grant connect, create session to "ORACLECLIENT@DOMAIN.COMPANY.COM";
```

Step 11. Login to the client machine via the Windows Domain client user (for MSLSA) or perform an okinit to authenticate the client Kerberos user (for file based CC):

```
okinit oracleclient
```

In the credential cache case, use `oklist` to verify there are valid credentials in the cache. If those credentials are expired, then use `okdstry` to clean the cache and then execute a new `okinit` command.

Step 12. Run the ODP.NET application

Set the ODP.NET connection string to User Id=/ and leave the Password blank.

Note

- After configuring the client and server, the last 2 steps are the only steps required on an ongoing basis to run the ODP.NET application.
- A Microsoft Visual C Run-Time Library (MSVCRT.DLL) bug can cause ODP.NET, Managed Driver's setting of the Kerberos5 configuration to be ignored by the Microsoft run-time. In such a case, you will encounter the error message:

```
OracleInternal.Network.NetworkException (0x80004005): NA Kerberos5:
Authentication handshake failure at stage: krb5_sname_to_principal: default
realm not found. Please set SQLNET.Kerberos5_conf.
```

To workaround this error, manually set `KRB5_CONFIG` in the ODP.NET application's run-time environment to point to the Kerberos5 configuration file pointed to by `SQLNET.Kerberos5_conf`. For example,

```
set KRB5_CONFIG=c:\oracle\network\admin\krb5.ini
```

See Also

Configuring Kerberos Authentication in *Oracle Database Security Guide* for more information about Kerberos and configuration with Oracle database.

Using Windows Native Authentication (NTS)

With the Windows native authentication adapter, Oracle users can authenticate to the database using just their Windows user login credentials. It provides a way to enable single sign-on and to simplify user and role credential management. Windows native authentication is also known as Windows Native authentication (NTS).

Note

- ODP.NET Core supports Windows Native Authentication on Windows only
- Due to a limitation in the Microsoft .NET APIs, ODP.NET, Managed Driver only supports Windows Native authentication (NTS) via Microsoft NT LAN Manager (NTLM) instead of Kerberos-based credentials. Normally, this limitation would be invisible to the ODP.NET, Managed Driver application, since the Windows domain and the Oracle database server will transparently support both NTLM and Kerberos domain credentials by default.

Configuring Windows Native Authentication (NTS) for the ODP.NET Client

Steps in configuring the NTS for the ODP.NET Client:

- [Step 1. Ensure OSAUTH_PREFIX_DOMAIN is set correctly](#)
- [Step 2. Setup the externally identified database user](#)
- [Step 3. Setup the client configuration to utilize NTS as the authentication methodology](#)

Step 1. Ensure OSAUTH_PREFIX_DOMAIN is set correctly

Make sure OSAUTH_PREFIX_DOMAIN is set appropriately. If you desire the externally identified user ID to include the domain, set it to true, otherwise false. The parameter is a registry setting that can be found at HKLM/software/oracle/HOME<ORACLE_SID>. For example, if your ORACLE_SID is r1, it is located at HKLM/software/oracle/HOMEr1.

Step 2. Setup the externally identified database user

If you set the parameter to true in Step 1, use the following commands to setup the externally identified database user associated with the desired Windows domain user:

```
create user "MYDOMAIN\MYUSER" identified externally;  
grant connect, create session to "MYDOMAIN\MYUSER";
```

Step 3. Setup the client configuration to utilize NTS as the authentication methodology

Edit the client sqlnet.ora or app config to add NTS to the sqlnet.authentication_services. For example.

```
sqlnet.authentication_services = (NTS)
```

Note

After configuring the client and server, the last 2 steps are the only steps required on an ongoing basis to run the ODP.NET application.

See Also

Authenticating Database Users with Windows in *Oracle Database Platform Guide for Microsoft Windows* for Windows for more information about Windows native authentication.

Operating System Authentication Credentials

Oracle Database can use Windows user login credentials to authenticate database users. To open a connection using Windows user login credentials, the User Id connection string attribute must be set to a slash (/). If the Password attribute is provided, it is ignored.

Note

ODP.NET Core supports operating system authentication for Windows only.

All ODP.NET, Unmanaged Driver connections, including those using operating system authentication, can be pooled. ODP.NET, Managed Driver supports operating system authentication, except when the Windows domain is constrained to only support Kerberos-based domain authentication. Connections are pooled by default, and no configuration is required, as long as pooling is enabled.

The following example shows the use of operating system authentication:

```
/* Create an OS-authenticated user in the database
   Assume init.ora has OS_AUTHENT_PREFIX set to "" and <OS_USER>
   is any valid OS or DOMAIN user.

       create user <OS_USER> identified externally;
       grant connect, resource to <OS_USER>;

   Login through OS Authentication and execute the sample. See Oracle
   documentation for details on how to configure an OS-Authenticated user
*/

// C#

using System;
using Oracle.DataAccess.Client;

class OSAuthenticationSample
{
    static void Main()
    {
        OracleConnection con = new OracleConnection();

        //Establish connection using OS Authentication
        con.ConnectionString = "User Id=/;Data Source=oracle;";
        con.Open();
        Console.WriteLine("Connected to Oracle" + con.ServerVersion);

        // Close and Dispose OracleConnection object
        con.Close();
        con.Dispose();
        Console.WriteLine("Disconnected");
    }
}
```

① See Also

Oracle AI Database Administrator's Reference for Microsoft Windows for information on how to set up Oracle Database to authenticate database users using Windows user login credentials

Network Data Encryption and Integrity

ODP.NET enables data encryption and integrity over a network for both intranet and cloud deployments. This ensures that data is disguised to all, except authorized users, and guarantees the original message contents are not altered. In earlier releases, these features were known as Oracle Advanced Security Option (ASO) encryption. Starting with Oracle Database 12c, Oracle ASO is not required to use network data encryption and data integrity.

The `SQLNET.ALLOW_WEAK_CRYPT0`, `OracleConfiguration.SqlNetAllowWeakCrypto`, or equivalent settings must be set to true (default) to use weaker encryption and checksum algorithms.

Using Data Encryption

All ODP.NET provider types support the following encryption standards and algorithms.

- Advanced Encryption Standard (AES)
 - AES 128-bit
 - AES 192-bit
 - AES 256-bit
 - Triple-DES (3DES)
 - 112-bit*
 - 168-bit*
- * = Encryption algorithms that are considered weak

ODP.NET, Managed Driver and ODP.NET Core use the following settings to configure network encryption:

- `SQLNET.ENCRYPTION_CLIENT`
- `SQLNET.ENCRYPTION_TYPES_CLIENT`

① See Also

[settings section](#) for definition and information on usage.

Using Data Integrity

All ODP.NET provider types support the following data integrity algorithms:

- SHA-1
- SHA-2

- SHA-256
- SHA-384
- SHA-512

📘 See Also

- For more information on network encryption and integrity or configuring them for ODP.NET, Unmanaged Driver, refer to the *Oracle Database Security Guide*.
- To configure network encryption or data integrity in ODP.NET, Managed Driver and ODP.NET Core, refer to the `SQLNET.CRYPTO_CHECKSUM_CLIENT` and `SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT` settings in [Oracle Data Provider for .NET, Managed Driver Configuration](#) or `sqlnet.ora`. On the database server machine, you will likely have to configure the `SQLNET.CRYPTO_CHECKSUM_SERVER` and `SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER` parameters in the `sqlnet.ora` file.

Schema Discovery

ADO.NET exposes five different types of metadata collections through the `OracleConnection.GetSchema` API. This permits application developers to customize metadata retrieval on an individual-application basis, for any Oracle data source. Thus, developers can build a generic set of code to manage metadata from multiple data sources.

The following types of metadata are exposed:

- `MetaDataCollections`
A list of metadata collections that is available from the data source, such as tables, columns, and indexes.
- `Restrictions`
The restrictions that apply to each metadata collection, restricting the scope of the requested schema information.
- `DataSourceInformation`
Information about the instance of the database that is currently being used, such as product name and version.
- `DataTypes`
A set of information about each data type that the database supports.
- `ReservedWords`
Reserved words for the Oracle query language.

📘 See Also

[Oracle Schema Collections](#)

User Customization of Metadata

ODP.NET provides a comprehensive set of database schema information. Developers can extend or customize the metadata that is returned by the `GetSchema` method on an individual application basis.

To do this, developers must create a customized metadata file and provide the file name to the application as follows:

1. Create a customized metadata file and put it in the `CONFIG` subdirectory where the .NET framework is installed. This is the directory that contains `machine.config` and the security configuration settings.

This file must contain the entire set of schema configuration information, not just the changes. Developers provide changes that modify the behavior of the schema retrieval to user-specific requirements. For instance, a developer can filter out internal database tables and just retrieve user-specific tables

2. Add an entry in the `app.config` file of the application, similar to the following, to provide the name of the metadata file, in name-value pair format.

```
<oracle.dataaccess.client>
  <settings>
    <add name="MetaDataXml" value="CustomMetaData.xml" />
  </settings>
</oracle.dataaccess.client>
```

When the `GetSchema` method is called, ODP.NET checks the `app.config` file for the name of the customized metadata XML file. First, the `GetSchema` method searches for an entry in the file with a element named after the provider, in this example, `oracle.dataaccess.client`. In this XML element, the value that corresponds to the name `MetaDataXml` is the name of the customized XML file, in this example, `CustomMetaData.xml`.

If the metadata file is not in the correct directory, then the application loads the default metadata XML file, which is part of ODP.NET.

 **See Also**
["GetSchema"](#)

Connection Pooling

ODP.NET connection pooling is enabled and disabled using the `Pooling` connection string attribute. By default, connection pooling is enabled. The following are `ConnectionString` attributes that control the behavior of the connection pooling service:

- `Connection Lifetime`
- `Connection Timeout`
- `Decr Pool Size`
- `HA Events`
- `Incr Pool Size`
- `Load Balancing`

- Max Pool Size
- Min Pool Size
- Pooling
- Validate Connection

Connection Pooling Example

The following example opens a connection using `ConnectionString` attributes related to connection pooling.

```
// C#

using System;
using Oracle.DataAccess.Client;

class ConnectionPoolingSample
{
    static void Main()
    {
        OracleConnection con = new OracleConnection();

        //Open a connection using ConnectionString attributes
        //related to connection pooling.
        con.ConnectionString =
            "User Id=scott;Password=tiger;Data Source=oracle;" +
            "Min Pool Size=10;Connection Lifetime=100000;Connection Timeout=60;" +
            "Incr Pool Size=5; Decr Pool Size=2";
        con.Open();
        Console.WriteLine("Connection pool successfully created");

        // Close and Dispose OracleConnection object
        con.Close();
        con.Dispose();
        Console.WriteLine("Connection is placed back into the pool.");
    }
}
```

Using Connection Pooling

When connection pooling is enabled (the default), the `Open` and `Close` methods of the `OracleConnection` object implicitly use the connection pooling service, which is responsible for pooling and returning connections to the application.

The connection pooling service creates connection pools by using the `ConnectionString` property as a signature, to uniquely identify a pool.

In managed and Core versions of ODP.NET, connection strings themselves must be an exact match in their entirety. If keywords are supplied in a different order or a space is added to the connection string, a new pool is created. If a pool already exists with the requested signature, a connection is returned to the application from that pool.

In unmanaged ODP.NET, a new pool is created only when connection string attribute values change. Extra spaces or changing keyword order do not create a new pool.

When a connection pool is created, the connection pooling service initially creates the number of connections defined by the `Min Pool Size` attribute of the `ConnectionString` property. This number of connections is always maintained by the connection pooling service for the connection pool, except when Fast Connection Failover removes invalid connections is

exceeded. In this case, the connection number could drop below the Min Pool Size. ODP.NET would then attempt to restore the minimum pool size level upon the next connection request.

At any given time, these connections are in use by the application or are available in the pool.

The Incr Pool Size attribute of the `ConnectionString` property defines the number of new connections to be created by the connection pooling service when more connections are needed in the connection pool.

When the application closes a connection, the connection pooling service determines whether or not the connection lifetime has exceeded the value of the `Connection Lifetime` attribute. If so, the connection pooling service destroys the connection; otherwise, the connection goes back to the connection pool. The connection pooling service enforces the `Connection Lifetime` only when `Close()` or `Dispose()` is invoked.

The Max Pool Size attribute of the `ConnectionString` property sets the maximum number of connections for a connection pool. If a new connection is requested, but no connections are available and the limit for Max Pool Size has been reached, then the connection pooling service waits for the time defined by the `Connection Timeout` attribute. If the `Connection Timeout` time has been reached, and there are still no connections available in the pool, the connection pooling service raises an exception indicating that the connection pool request has timed-out. Upon a connection timeout, ODP.NET distinguishes whether the timeout occurred due to the database server failing to deliver a connection in the allotted time or no connection being available in the pool due to the maximum pool size having been reached. The exception text returned will either be "Connection request timed out" in the case of the former or "Pooled connection request timed out" in the case of the latter.

The `Validate Connection` attribute validates connections coming out of the pool. This attribute should be used only when absolutely necessary, because it causes a round-trip to the database to validate each connection immediately before it is provided to the application. If invalid connections are uncommon, developers can create their own exception/error handler to retrieve and validate a new connection, rather than using the `Validate Connection` attribute. This generally provides better performance.

The connection pooling service closes connections when they are not used; connections are closed every 3 minutes. The `Decr Pool Size` attribute of the `ConnectionString` property provides connection pooling service for the maximum number of connections that can be closed every 3 minutes.

Enabling connection pooling by setting "pooling=true" in the connection string (which is the case by default) will also pool operating system authenticated connections.

Connection Pool Management

ODP.NET connection pool management provides explicit connection pool control to ODP.NET applications. Applications can explicitly clear connections in a connection pool.

Using connection pool management, applications can do the following:

Note

- Clear connections from connection pools using the `ClearPool` method.
- Clear connections in all the connection pools in an application domain, using the `ClearAllPools` method.

① See Also

- ["ClearPool"](#)
- ["ClearAllPools"](#)

Connection Performance Counters

Installing Oracle Data Provider for .NET creates a set of performance counters on the target system. All ODP.NET provider types (Core, managed, and unmanaged) publish these performance counters for each of their client applications. These performance counters can be viewed using Windows Performance Monitor (Perfmon) or the `PerformanceCounter` class in the `System.Diagnostics` namespace version 4.5.0 or higher.

① Note

ODP.NET performance counters are available on Windows platforms only.

As ODP.NET performance counters are not enabled nor registered after installation, administrators must register the counters, then enable the specific counters of interest before being able to monitor them.

Registering Performance Counters

After installation, ODP.NET performance counters must be registered so that tools, such as Perfmon, can consume them. Registration requires running a PowerShell script for the respective provider.

ODP.NET installs six scripts for counters that must be run from PowerShell to take effect

- `register_odpc_perfmon_counters.ps1` – Registers ODP.NET Core counters
- `register_odpm_perfmon_counters.ps1` – Registers managed ODP.NET counters
- `register_odpu_perfmon_counters.ps1` – Registers unmanaged ODP.NET counters
- `unregister_odpc_perfmon_counters.ps1` – Unregisters ODP.NET Core counters
- `unregister_odpm_perfmon_counters.ps1` – Unregisters managed ODP.NET counters
- `unregister_odpu_perfmon_counters.ps1` – Unregisters unmanaged ODP.NET counters

These scripts are located in the `ORACLE_HOME\odp.net\PerfCounters` directory for Oracle Universal Installer and NuGet deployments, in the `<INSTALLATION_DIRECTORY>\odp.net\PerfCounters` directory for xcopy deployments, and `<VISUAL_STUDIO_SOLUTION_DIRECTORY>\packages\Oracle.ManagedDataAccess.<VERSION>\PerfCounters` directory for MSI deployments.

Windows Administrator privileges are required when running these PowerShell scripts.

To unregister, run the unregister PowerShell script that matches the ODP.NET provider type you wish to unregister.

Previously, the `OraProvCfg.exe` utility was included with ODP.NET for registering and unregistering counters. These PowerShell scripts replace the utility. If you registered ODP.NET counters with this utility, then unregister with the same `OraProvCfg.exe` version.

Enabling Performance Counters

ODP.NET enables monitoring many different connection counters, including pooled and non-pooled connections. These counters can be monitored individually or together. Developers can set which counters to monitor prior to application startup using the `PerformanceCounters` setting in the `<settings>` section of the .NET config file, which is available in managed and unmanaged ODP.NET, or Windows Registry, which is available in unmanaged ODP.NET only. For ODP.NET Core, developers can set the `OracleConfiguration.PerformanceCounters` property.

[Table 3-7](#) lists the connection performance counters with their setting values.

Table 3-7 Performance Counters for Connection Pooling

Performance Counter	Valid Values	Description
None	0	Not enabled (Default)
HardConnectsPerSecond	1	Number of sessions being established with the Oracle Database every second.
HardDisconnectsPerSecond	2	Number of sessions being severed from the Oracle Database every second.
SoftConnectsPerSecond	4	Number of active connections originating from connection pools every second.
SoftDisconnectsPerSecond	8	Number of active connections going back to the connection pool every second.
NumberOfActiveConnectionPools	16	Total number of active connection pools.
NumberOfInactiveConnectionPools	32	Number of inactive connection pools.
NumberOfActiveConnections	64	Total number of connections in use.
NumberOfFreeConnections	128	Total number of connections available for use in all the connection pools.
NumberOfPooledConnections	256	Total number of pooled active and free connections.
NumberOfNonPooledConnections	512	Number of non-pooled active connections.
NumberOfReclaimedConnections	1024	Number of connections which were garbage-collected implicitly.
NumberOfStasisConnections	2048	No operation. This counter is no longer supported.

These settings are bitwise ORed in order to monitor more than one counter. For example, to collect counters for SoftConnectsPerSecond (4), Soft DisconnectsPerSecond (8), and NumberOfActiveConnectionPools (16), set PerformanceCounters to 28 (i.e. 4 + 8 + 12).

Setting Performance Counters in .NET Configuration Files

Performance counters can be set using an .NET configuration file, such as web.config or app.config, in managed and unmanaged ODP.NET only. Since .NET configuration entries take precedence over Windows Registry settings, they can be used for a specific application.

As earlier described, populating the PerformanceCounters setting in the <settings> section enables individual performance counters to be monitored. Windows Perfmon uses very long and not easily human readable application instance names. To make identifying each application easier administrators can assign an identifier name in the <connectionPools> section of the .NET configuration file. This section supports the following settings:

- **connectionString:** This setting identifies the connections or pool to monitor using the connection string as the unique identifier. The connection string entered here must match the target connection string to be monitored, but without the password attribute.
- **poolName (optional):** Connection strings can be very long and hard to read with many that look similar. poolName allows using an arbitrary string to easily identify which pool is being monitored.

The following examples show how to use these settings.

ODP.NET, Managed Driver

```
<oracle.manageddataaccess.client>
  <version number="*">
    <connectionPools>
      .
      .
      <connectionPool connectionString="[connection string without password]"
poolName="[Pool Name]"> </connectionPool>
      .
    </connectionPools>
  </version>
</oracle.manageddataaccess.client>
```

ODP.NET, Unmanaged Driver can use the same pool name setting and format as listed above by replacing the <oracle.manageddataaccess.client> tags with <oracle.unmanageddataaccess.client> tags. Alternatively, the legacy <oracle.dataaccess.client> format is available for unmanaged ODP.NET.

ODP.NET, Unmanaged Driver

```
<configuration>
  <oracle.dataaccess.client>
    <settings>
      .
      .
      <add name="[connection string without password]" value="connectionPool name='[Pool
Name]'" />
      .
    </settings>
  </oracle.dataaccess.client>
</configuration>
```

Setting Performance Counters in Windows Registry

Publication of individual unmanaged ODP.NET performance counters is enabled or disabled using the Windows Registry value PerformanceCounters of type REG_SZ. This registry value is under:

HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\ODP.NET\Assembly_Version

where *Assembly_Version* is the full assembly version number of Oracle.DataAccess.dll.

Similarly to the .NET config file, multiple performance counters can be obtained by adding the valid values. For example, if PerformanceCounters is set to 3, then both HardConnectsPerSecond and HardDisconnectsPerSecond are enabled.

Publishing Performance Counters

Commonly, ODP.NET performance counters are monitored using Perfmon. The counters are published under the following Category Names:

- ODP.NET, Core Driver
- ODP.NET, Managed Driver
- ODP.NET, Unmanaged Driver

Administrators can choose the individual ODP.NET counters to monitor after selecting one or more of these categories. PerfMon shows all ODP.NET counters, but only the explicitly enabled counters generate statistics.

After choosing the counters to monitor, administrators then select the running instance(s) to monitor. ODP.NET instances must be actively running for them to appear in PerfMon. Otherwise, no instances will appear available to monitor. After instance selection occurs, they are added to PerfMon as counters to monitor.

Performance counters can monitor at the application domain, pool, or database instance level. Database instance level monitoring only applies if load balancing or Fast Connection Failover features are enabled.

The instance name format is as follows:

<Application Domain Name> [<Process Id>, <Application Domain Id>][<Connection String/Pool Name>][<Instance Name>]. The entry is limited to 127 characters. There is a restriction length on every field in the instance name. The following table shows the maximum number of characters allocated for each field:

Table 3-8 Field Names of Performance Counters and Maximum Number of Characters

Field Name	Maximum Number of Characters
Application Domain	40
Pool Name/Connection String	70
Database Instance Name	16

When the length of a field value exceeds the length limit, the string is truncated and appended with ". . ." to fit within the length limit and indicate the continuation. For example, for a given application called Program.exe with a connection string user id=hr;Password=<password>;data source=inst1;max pool size=125;min pool size=50,

one may see the following similar to the following for a process that has two application domains:

- Program.exe [123, 1]
- Program.exe [123, 1][user id=hr;data source=inst1;max pool siz...]
- Program.exe [123, 1][user id=hr;data source=inst1;max pool siz...] [instA]
- Domain 2[123, 2]
- Domain 2[123, 2][user id=hr;data source=inst1;max pool siz...]
- Domain 2[123, 2][user id=hr;data source=inst1;max pool siz...] [instB]
- Domain 2[123, 2][user id=hr;data source=inst1;max pool siz...] [instC]

Since connection pool attributes can be similar in their first 70 characters, applications can set a Pool Name to uniquely identify each one in the monitoring tool. For example, when using Pool Name, the process will show up as follows:

Domain 2[123, 2][Pool Name][instC]

Database Resident Connection Pooling and Connection Manager in Traffic Director Mode

Client side connection pooling can be very efficient for middle tier machines. However, it can consume a great deal of database server resources if there are numerous middle tier servers with idle connections. While the intent is to keep the number of idle connections to a minimum, it becomes difficult as the number of middle tier servers increase, each possibly having idle connections that cannot be shared across applications.

Database Resident Connection Pooling (DRCP) is intended to optimize resource usage by pooling connections at the database server level, which can then be shared across many applications. The benefit is better scalability and lower resource usage at the database server level.

Oracle's database proxy solution, Connection Manager in Traffic Director Mode (CMAN-TDM) has its own pooling feature, Proxy Resident Connection Pooling (PRCP). PRCP works similarly to DRCP. If an application works well with DRCP, it will work just as well with PRCP. The only change necessary on the application side is the TNS alias or Easy Connect string should point to the PRCP server instead of the database/DRCP server.

ODP.NET supports DRCP and PRCP, which allows dispensing and releasing these connections to better utilize database server resources.

About DRCP

DRCP pools server processes, each of which is the equivalent to a dedicated server process and database session combined. These are called pooled servers. Pooled servers can be shared by multiple applications running on the same or multiple hosts.

When DRCP is configured for Oracle Database Real Application Clusters (Oracle RAC), the pool configuration is applied to each database instance. Starting or stopping the pool on one instance starts or stops the pool on all instances.

DRCP and ODP.NET

DRCP is a server side pool that complements ODP.NET client side pooling. These two pools can be used together.

In a typical dedicated server mode, client side connection pooling saves both on server round trips and socket/session creation. In a connection open/close sequence, a server round trip is involved only upon the first `Open()`. For subsequent open/close sequences on the same connection, no client to server interaction is required because the connection is pooled locally in the client address space.

With DRCP enabled, when the client creates a connection, the connection is assigned a pooled server when needed. When the connection is closed, the database server releases the DRCP server session back to the server side pool, effectively making the DRCP server session available for reuse. This server session can be reused by the same pool or a different pool on the same or different middle tier hosts.

Configuring DRCP and ODP.NET

The following section describe how DRCP can be configured on the client side and the server side.

- Configuring DRCP on the Client Side

ODP.NET developers can enable DRCP on the client side by using `(SERVER=POOLED)` in the connect descriptor.

Sample Descriptor:

```
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=myhost)(PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=sales.example.com)(SERVER=POOLED)))
```

- Configuring DRCP on the Server Side

Oracle database includes a default connection pool called `SYS_DEFAULT_CONNECTION_POOL`. By default, this pool is created, but not started. To enable database resident connection pooling, you must explicitly start the connection pool.

You must be a database administrator (DBA) and must log on as SYSDBA to start and end a pool.

```
SQL> EXECUTE DBMS_CONNECTION_POOL.START_POOL();
```

For DRCP connections to be shared across multiple client-side ODP.NET connection pools, then set the `OracleConnection.DRCPConnectionClass` property to a string value before opening the ODP.NET connection. ODP.NET will first try to obtain an idle connection with the same DRCP connection class property value. If it does not find one, then it will create a new connection instead.

Example 3-1 Using Database Resident Connection Pooling: Sample Code

```
// This application uses the following connect descriptor:
// oracle = (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=<hostname>)(PORT=1521))
// (CONNECT_DATA=(SERVICE_NAME=<service name>)(SERVER=POOLED)))

using System;
using Oracle.ManagedDataAccess.Client;

class DRCP
{
    static void Main()
    {
        string constr = "user id=hr;password=hr;data source=oracle";
```

```
        OracleConnection con = new OracleConnection(constr);
        con.DRCPConnectionClass = "GroupA";
        con.Open();

        con.Dispose();
    }
}
```

① Note

To use DRCP with ODP.NET, Unmanaged Driver,:

- the .NET configuration setting, CPVersion, must be set to 2.0, or,
- have CPVersion not set at all, but have (SERVER=POOLED) in the TNS full descriptor that is used by the application.

Refer to [settings section](#) for more information.

① See Also

Oracle Database Administrator's Guide for more information on configuring the default connection pool or end pool.

DRCP for Oracle Multitenant

In Oracle Database 21c and higher, each PDB can have its own separate DRCP.

In Oracle Database 19c and earlier, a DRCP can only be created on the CDB level and then be shared among all its PDBs. Thus, ODP.NET would have one DRCP for the entire CDB that would be configured and managed (start, stop, or reconfigure) only by connecting to the root container.

To configure, set a session to point to the CDB and start the DRCP pool. For example:

```
alter session set container = cdb$root;
execute dbms_connection_pool.configure_pool('SYS_DEFAULT_CONNECTION_POOL');
execute dbms_connection_pool.start_pool();
```

DRCP for Oracle Real Application Clusters (RAC)

In an Oracle Real Application Clusters (RAC) environment, when a user creates a DRCP on an instance, that DRCP is replicated on all the instances of an Oracle RAC database and you can use any instance to manage the connection pool. Any changes you make to the pool configuration are applicable on all Oracle RAC instances.

DRCP Restrictions for ODP.NET

ODP.NET does not support the following features with DRCP:

- Open the proxy connection with end user authentication
- Open the sysoper connection

- Open using `OpenWithNewPassword`
- Set the `SQLNET.AUTHENTICATION_SERVICES` parameter value to `nts`, which enables Microsoft Windows native operating system authentication
- .NET Framework versions earlier than .NET Framework 4

Oracle Multitenant and Pluggable Databases

Oracle Database 12c introduced Oracle Multitenant, which enables an Oracle database to contain a portable collection of schemas, schema objects, and nonschema objects that appears to ODP.NET as a separate database. This self-contained collection is called a pluggable database (PDB).

Oracle Multitenant is a database architecture that enables customers to easily consolidate multiple pluggable databases without changing their application. This architecture delivers all the benefits of managing many databases as one, yet retains the isolation and resource prioritization of separate databases. In addition, Oracle Multitenant enables rapid provisioning and upgrades, and fully complements other Oracle database options.

Managed and unmanaged ODP.NET fully support Oracle Multitenant. In addition to being able to connect to individual pluggable databases, ODP.NET has been enhanced so that applications can request and obtain connections to specific PDBs from the same connection pool. The application can connect to any of the PDBs that are part of the same container database (CDB) and they must have the same common user. Moreover, if connections span multiple instances, such as with Oracle Real Application Clusters, then every instance must uniformly have the same exact PDBs and services for the common user to connect to. If these conditions are met, then ODP.NET will be able to connect to or implicitly switch connections from one PDB to another within the same pool in a highly performant manner.

To obtain a connection to a specific PDB (as a common user) set the `OracleConnections` properties, `PDBName` and `ServiceName`, and then call `OracleConnection.Open()` to obtain the a connection to the specified PDB and Service name. `PDBName` and `ServiceName` identify the PDB and service that the connection needs to be established to or switched to if using an existing idle connection from the connection pool. They are the pluggable database name and database service name, respectively. If only the `PDBName` property is set and the service name is not set by the user, then the provider will return a connection that uses the default (administrative) service.

Note

For ODP.NET, Unmanaged Driver to be able to switch PDB connections within the same pool:

- the .NET configuration setting, `CPVersion`, must be set to 2.0, or,
- have `CPVersion` not set at all, but set `PDBName` and/or `ServiceName` to a non-null/non-empty value for the first connection request for a given connection string.

Refer to [settings section](#) for more information.

Example 3-2 Using Pluggable Database: Sample Code

```
// C#
using System;
using Oracle.ManagedDataAccess.Client;
```

```
class PDB
{
    static void Main()
    {
        string constr = "user id=hr;password=hr;data source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.PDBName = "pdb1";
        con.ServiceName = "db1.company.com";
        con.Open();
        con.Close();
    }
}
```

ODP.NET keeps track of the PDBName and ServiceName to which the pooled connections are established to. Upon the application requesting for a connection with a specified PDBName and ServiceName, ODP.NET will return a connections that matches that request. However, if a matching connection is not found, ODP.NET will create a new connection and/or alter the session to switch to the requested PDBName and ServiceName. If the PDBName and/or ServiceName has been altered, then the SwitchedConnection property will return true if called with the Open() method invocation.

ODP.NET does not support usage of the ALTER SESSION statement to modify the container database during the lifetime of a process. If using PDBs with Oracle Continuous Query Notification, you must connect to Oracle Database 12c Release 2 or higher. Hosting connections to multiple PDBs from the same pool requires ODP.NET for .NET Framework 4 or higher.

① Note

When the connection is implicitly being switched from one PDB/Service to another, Service Relocation Connection Timeout(SRCT) will not take effect.

① See Also

Managing Pluggable Databases in *Oracle Database Administrator's Guide*

Edition-Based Redefinition

Edition-based redefinition enables you to upgrade the database component of an application even while the application is being used. This minimizes or eliminates downtime for the application.

ODP.NET does not support usage of the ALTER SESSION statement to modify the Edition during the lifetime of a process.

Managed and unmanaged ODP.NET can connect to a database edition, and also have the ability to host connections to multiple Editions from within the same pool. An ODP.NET application can obtain a connection that is associated with a specified database edition by setting the OracleConnection.DatabaseEditionName property to the name of the database edition that the connection should be associated with. To use a single pool, the same user must be able to connect to both Editions. Applications will perform better and more efficiently

when using Editions because ODP.NET can use the same connection pool and even share the same `OracleConnection` object by only changing Edition-specific properties.

Example 3-3 Using Edition-Based Redefinition: Sample Code

```
using System;
using Oracle.ManagedDataAccess.Client;

class Editions
{
    static void Main()
    {
        // Create a connection
        string constr = "user id=hr;password=hr;data source=oracle";
        OracleConnection con = new OracleConnection(constr);

        // Obtain a connection associated with EditionX
        con.DatabaseEditionName = "EditionX";
        con.Open();

        // Obtain a connection associated with EditionY
        con.Close();
        con.DatabaseEditionName = "EditionY";
        con.Open();
        con.Dispose();
    }
}
```

Changing the edition name through `DatabaseEditionName` while the connection is open is not allowed. It must be changed while the connection is in a closed state.

ODP.NET supports Editions when connecting to Oracle Database 11g Release 2 or later. Hosting multiple Editions in the same pool is available only in ODP.NET for .NET Framework 4 and above.

Note

To use this Edition-Based Redefinition feature with unmanaged ODP.NET connection pools:

- the .NET configuration setting, `CPVersion`, must be set to 2.0, or,
- have `CPVersion` not set at all, but set the `DatabaseEditionName` property to a non-null/non-empty value for the first connection request for a given connection string.

Refer to [settings section](#) for more information.

Applications can specify an Edition at deployment time using the registry or configuration file. An application can create the following registry entry of type `REG_SZ`:

`HKLM\Software\Oracle\ODP.NET\version\Edition`

Here *version* is the version of ODP.NET, and *Edition* is a valid Edition string value.

An application can alternatively use the `web.config` or `application.config` configuration file to specify the Edition at deployment time. The `machine.config` configuration file can be used to specify the Edition for all applications that use a particular version of the .NET framework.

The following example sets the Edition to E1 in a .NET configuration file for ODP.NET, Unmanaged Driver:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <oracle.dataaccess.client>
    <settings>
      <add name="Edition" value="E1"/>
    </settings>
  </oracle.dataaccess.client>
</configuration>
```

To summarize, the Edition name can be set by the `DatabaseEditionName` property, in the .NET configuration file, or in the Windows Registry. If `DatabaseEditionName` is set, then it takes precedence over the other two settings. If the .NET configuration file has an Edition set, then it takes precedence over the registry.

See Also

For more information on Editions refer to the *Oracle AI Database Administrator's Guide* and *Oracle AI Database Development Guide*

Privileged Connections

Oracle allows database administrators to connect to Oracle Database with various privileges, such as SYSDBA, SYSASM, and SYSOPER. This is done through the DBA Privilege attribute of the `ConnectionString` property.

The following example connects `scott/tiger` as SYSDBA:

```
// C#

using System;
using Oracle.DataAccess.Client;

class PrivilegedConnectionSample
{
    static void Main()
    {
        OracleConnection con = new OracleConnection();

        //Connect scott/tiger as SYSDBA
        con.ConnectionString = "User Id=scott;Password=tiger;" +
            "DBA Privilege=SYSDBA;Data Source=oracle;";
        con.Open();
        Console.WriteLine("Connected to Oracle" + con.ServerVersion);

        // Close and Dispose OracleConnection object
        con.Close();
        con.Dispose();
        Console.WriteLine("Disconnected");
    }
}
```

① See Also

DBA Privilege "[Table 6-51](#)" for further information on privileged connections in the database

Connection Pooling with OracleCredential

To better secure passwords, all ODP.NET applications can use `OracleCredential` to store user names and passwords outside of the connection string. `OracleCredential` mitigates the possibility of exposing user credentials in a page file swap or in a crash dump.

`OracleCredential` can be used with or without connection pooling.

When using the `OracleCredential` object, one should be aware of the connection pool algorithm that has been modified to take this new feature into consideration, when opening connection to the Oracle database. With earlier implementations of ODP.NET connection pool algorithm, the uniqueness of connection string attribute values were used to decide if a new connection pool needs to be created or an existing connection pool can be used to dispense connections. But with the introduction of `OracleCredential`, the reference of this object is also used to decide which connection pool that `OracleConnection` object is associated with.

The decision to create a new connection pool now depends on two factors, first is the uniqueness of connection string (as in earlier releases) and the second is the reference comparison of the `OracleCredential` provided to the `OracleConnection` object while opening a new connection. If either of these is different, a separate pool is used. In order to keep re-using the same connection pool and not create additional connection pools while using the `OracleCredential` object, simply create one `OracleCredential` object per database user and re-use the same object when opening connections.

Here is an example to clarify the new algorithm:

```
using System;
using System.Security;
using Oracle.ManagedDataAccess.Client;
//using Oracle.DataAccess.Client;

class Test
{
    static void Main()
    {
        string connStr1 = "user id=hr; password=hr; data source=oracle";
        string connStr2 = "data source=oracle";

        SecureString secPwd = new SecureString();
        secPwd.AppendChar('h');
        secPwd.AppendChar('r');
        secPwd.MakeReadOnly();

        OracleCredential oc1 = new OracleCredential("hr", secPwd);
        OracleCredential oc2 = new OracleCredential("hr", secPwd);
        OracleCredential oc3 = new OracleCredential("hr", secPwd);

        // con1 and con2 are associated with the same connection pool since both are using
        the same
        // connection string and OracleCredential remains null in both cases.
        OracleConnection con1 = new OracleConnection(connStr1);
```

```
OracleConnection con2 = new OracleConnection(connStr1, null);

// con3 and con4 use the same connection string but different OracleCredential
objects
// (although same contents) so they are associated with different connection pools.
OracleConnection con3 = new OracleConnection(connStr2, oc1);
OracleConnection con4 = new OracleConnection(connStr2, oc2);

// con5 and con6 use the same connection string and same OracleCredential object
// so they are associated with the same connection pool.
OracleConnection con5 = new OracleConnection(connStr2, oc3);
OracleConnection con6 = new OracleConnection(connStr2, oc3);

// Open the connections
con1.Open();
con2.Open();
con3.Open();
con4.Open();
con5.Open();
con6.Open();

// Please note that con1 and con2 use the same connection pool.
// In addition, con3 and con4 use different connection pools
// And lastly con5 and con6 use the same connection pool.
// Thus, in the end, there will be 4 different connection pools created in total.
}
}
```

Note

OracleCredential does not support double quotes around a SecureString password. Double quotes can be used within a password, however.

See Also

[OracleCredential Class](#)

Securing Passwords with OracleOpaqueString

Passwords can be stored as a .NET SecureString. However, the data type has weaknesses in keeping the text confidential. OracleOpaqueString better obfuscates string data without SecureString's limitations.

Oracle recommends using OracleOpaqueString for more robust security instead of SecureString.

Existing ODP.NET APIs that contain a SecureString parameter have a matching overloaded API that uses OracleOpaqueString as well. ODP.NET developers have the flexibility to use either SecureString or the more secure OracleOpaqueString.

Apps populate an empty OracleOpaqueString instance one character at a time. To use the data type, apps then make it read-only and immutable. This immutable state is permanent. Once the OracleOpaqueString is read-only, it can be used by ODP.NET to create a connection. This behavior matches that of SecureString.

One programmatic difference between `SecureString` and `OracleOpaqueString` is usage with the `OracleCredential` class. In order to set the Password or Proxy Password using an `OracleOpaqueString`, use one of the existing constructors with passwords set as null. Then, set the passwords using `SetPassword()` and `SetProxyPassword()` methods.

If an app wants to set or update a proxy user and password in an `OracleCredential`, it can only be done using `OracleOpaqueString` and the `SetPassword()` and `SetProxyPassword()` methods.

A benefit of the `SetPassword` and `SetProxyPassword` methods are that they can update an existing `OracleCredential` instance's passwords anytime the credentials change. This is only possible when using `OracleOpaqueString` and `OracleCredential`.

The following code sample shows how to use an `OracleOpaqueString` password with ODP.NET user credentials:

```
//Create OracleOpaqueString and populate it with the password
OracleOpaqueString opaquePassword = new OracleOpaqueString();
foreach (char c in "<PASSWORD>")
    opaquePassword.AppendChar(c);
opaquePassword.MakeReadOnly();

//Create OracleCredential and set the instance's values
OracleCredential cred = new OracleCredential("HR, null, OracleDBAPrivilege.NONE);
cred.SetPassword(opaquePassword);

//Create the connection, populate the connection string with the OracleCredential
values, and open
OracleConnection conn = new OracleConnection("data source=<DATA SOURCE>");
conn.Credential = cred;
conn.Open();
```

See Also

- [OracleOpaqueString Class](#)
- [OracleCredential Class](#)
- [OracleConnection Class](#)

Password Expiration

Oracle allows users passwords to expire. ODP.NET lets applications handle the password expiration by providing a new method, `OpenWithNewPassword`, that opens the connection with a new password.

The following example uses the `OracleConnection` `OpenWithNewPassword` method to connect with a new password of panther:

```
/* Database Setup
connect / as sysdba;
drop user testexpire cascade;
-- create user "testexpire" with password "testexpire"
grant connect , resource to testexpire identified by testexpire;
alter user testexpire password expire;
*/
```

```
// C#

using System;
using Oracle.DataAccess.Client;

class PasswordExpirationSample
{
    static void Main()
    {
        OracleConnection con = new OracleConnection();

        try
        {
            con.ConnectionString =
                "User Id=testexpire;Password=testexpire;Data Source=oracle";
            con.Open();
            Console.WriteLine("Connected to Oracle" + con.ServerVersion);
        }
        catch (OracleException ex)
        {
            Console.WriteLine(ex.Message);

            //check the error number
            //ORA-28001 : the password has expired
            if (ex.Number == 28001)
            {
                Console.WriteLine("\nChanging password to panther");
                con.OpenWithNewPassword("panther");
                Console.WriteLine("Connected with new password.");
            }
        }
        finally
        {
            // Close and Dispose OracleConnection object
            con.Close();
            con.Dispose();
            Console.WriteLine("Disconnected");
        }
    }
}
```

Note

- The `OpenWithNewPassword` method should be used only when the user password has expired, not for changing the password.
- If connection pooling is enabled, then invoking the `OpenWithNewPassword` method also clears the connection pool. This closes all idle connections created with the old password.

See Also

["OpenWithNewPassword\(String\)"](#)

Proxy Authentication

With proper setup in the database, proxy authentication enables middle-tier applications to control the security by preserving database user identities and privileges, and auditing actions taken on behalf of these users. This is accomplished by creating and using a proxy database user that connects and authenticates against the database on behalf of a database user (that is, the *real* user) or database users.

Proxy authentication can then be used to provide better scalability with connection pooling. When connection pooling is used in conjunction with proxy authentication, the proxy authenticated connections can be shared among different real users. This is because only the connection and session established for the proxy is cached. An additional session is created for the real user when a proxy authenticated connection is requested, but it will be destroyed appropriately when the proxy authenticated connection is placed back into the pool. This design enables the application to scale well without sacrificing security.

ODP.NET applications can use proxy authentication by setting the "Proxy User Id" and "Proxy Password" attributes in the connection string. The real user is specified by the "User Id" attribute. Optionally, to enforce greater security, the real user's password can be provided through the "Password" connection string attribute. When using distributed transactions in conjunction with proxy authentication, the real user's password is no longer optional, and it must be supplied.

The following example illustrates the use of ODP.NET proxy authentication:

```
/* Log on as DBA (SYS or SYSTEM) that has CREATE USER privilege.
   Create a proxy user and modified scott to allow proxy connection.

   create user appserver identified by eagle;
   grant connect, resource to appserver;
   alter user scott grant connect through appserver;
*/

// C#

using System;
using Oracle.DataAccess.Client;

class ProxyAuthenticationSample
{
    static void Main()
    {
        OracleConnection con = new OracleConnection();

        // Connecting using proxy authentication
        con.ConnectionString = "User Id=scott;Password=tiger;" +
            "Data Source=oracle;Proxy User Id=appserver;Proxy Password=eagle; ";
        con.Open();
        Console.WriteLine("Connected to Oracle" + con.ServerVersion);

        // Close and Dispose OracleConnection object
        con.Close();
        con.Dispose();
        Console.WriteLine("Disconnected");
    }
}
```

ODP.NET proxy authentication can also use the connection string syntax: `User Id=<PROXY USER ID>[<REAL USER ID>];Password=<PROXY PASSWORD>` in managed ODP.NET and

ODP.NET Core. For example, if we use the same credentials as the above proxy user code sample, `User Id=appserver[scott];Password=eagle`. This format applies to single session proxy connections. This format does not support two session proxy connections. ODP.NET supports this format when stored in SEPS wallets as well.

❏ See Also

- *Oracle AI Database SQL Language Reference* for the description and syntax of the proxy clause for the ALTER USER statement
- *Oracle AI Database Security Guide* for information about how auditing works for SQL statements and privileges in a multitier environment

Dynamic Distributed Transaction Enlistment

For those applications that dynamically enlist in distributed transactions through the `EnlistDistributedTransaction` of the `OracleConnection` object, the "Enlist" connection string attribute must be set to a value of "true". If "Enlist=true", the connection enlists in a transaction when the `Open` method is called on the `OracleConnection` object, if it is within the context of a COM+ transaction or a `System.Transactions`. If not, the `OracleConnection` object does not enlist in a distributed transaction, but it can later enlist explicitly using the `EnlistDistributedTransaction` or the `EnlistTransaction` method. If "Enlist" is equal to "false" or "dynamic", the connection cannot enlist in the transaction. ODP.NET, Unmanaged Driver in ODAC 12c Release 3 first introduced this new behavior for "Enlist=dynamic".

❏ See Also

- ["Connection String Attributes and URLs"](#)
- ["EnlistDistributedTransaction"](#)

Client Identifier and End-to-End Tracing

The client identifier is a predefined attribute from the Oracle application context namespace `USERENV`. It is similar to proxy authentication because it can enable tracking of user identities. However, client identifier does not require the creation of two sessions (one for the proxy user and another for the end user) as proxy authentication does. In addition, the client identifier does not have to be a database user. It can be set to any string. But most importantly, by using client identifier, ODP.NET developers can use application context and Oracle Label Security, and configure Oracle Virtual Private Database (VPD) more easily. To set the client identifier, ODP.NET applications can set the `ClientId` property on the `OracleConnection` object after opening a connection. If connection pooling is enabled in unmanaged ODP.NET, then the `ClientId` is reset to null whenever a connection is placed back into the pool. In managed ODP.NET, the `ClientId` is reset to null (assuming no new value is provided) upon the first database round trip after a connection is placed back into the pool.

The client identifier can also be used for end-to-end application tracing. End-to-end tracing simplifies the process of diagnosing performance problems in multitier environments. In multitier environments, a request from an end client is routed to different database sessions by the middle tier making it difficult to track a client across different database sessions. End-to-

end tracing uses the client identifier to uniquely trace a specific end-client through all tiers to the database server.

ODP.NET exposes the `ActionName`, `ClientId`, `ClientInfo`, and `ModuleName` write-only properties on the `OracleConnection` object. These properties correspond to the following end-to-end tracing attributes:

- `ActionName` - Specifies an action, such as an INSERT or UPDATE operation, in a module
- `ClientId` - Specifies an end user based on the logon ID, such as HR.HR
- `ClientInfo` - Specifies user session information
- `ModuleName` - Specifies a functional block, such as Accounts Receivable or General Ledger, of an application

📘 See Also

- ["OracleConnection Properties"](#)
- *Oracle AI Database SQL Tuning Guide* for an overview of End-to-End Application Tracing
- *Oracle Database Security Guide*

Automatic Connection Recovery

ODP.NET can detect invalid connections that occur through planned or unplanned outages, such as database failovers and network failures, at the start of a request boundary. When an invalid session is detected, ODP.NET fails over by creating a new session to any remaining database instance. The end user experiences no error since the invalid connection is not dispensed.

Automatic Connection Recovery (ACR) is enabled automatically starting in ODP.NET 23.8. It is available in all ODP.NET provider types (core, managed, and unmanaged) and requires no code nor configuration changes on the client nor database server side to use.

ACR is a different high availability feature than Application Continuity (AC), Fast Connection Failover (FCF), and Transparent Application Failover (TAF). ACR's primary goal is ensuring new requests receive valid connections. It does not recover a connection's session and transaction state and return it to the same end user after a failover as AC and TAF do.

ACR has the most functional similarity to FCF among the three. But unlike FCF, ACR checks the connection's session state, one by one, as each is dispensed. FCF sends connection and database status messages in real-time and removes invalid connections altogether once the failure message is received. FCF is a more responsive and faster approach.

ACR has the benefit that it requires no setup to use and is always on.

Reset Session State to Limit Application State Leaks

The `RESET_STATE` database service attribute partially clears ODP.NET session state when set to the value `LEVEL1`. It executes at the end of a request before processing the next request to limit session state leaks from one session to the next. The `RESET_STATE` service attribute is

recommended for all stateless applications to prevent the leakage of a session state to later reuses.

No ODP.NET code changes are required to use this feature. All ODP.NET provider types support this feature.

RESET_STATE does not reset all session state attributes.

Note

About Application Continuity to learn more about RESET_STATE and the session state attributes it resets.

Real Application Clusters and Global Data Services

This section discusses optimizations for the following products:

- Oracle Real Application Clusters (Oracle RAC) is a cluster database with a shared cache architecture that overcomes the limitations of traditional shared-nothing and shared-disk approaches to provide highly scalable and available database solutions for business applications.
- Oracle Data Guard provides one or more standby databases to protect Oracle data from failures, disasters, human error, and data corruptions for high availability in mission critical applications.
- Oracle GoldenGate replicates data among heterogeneous data environments. It enables high availability solutions, real-time data integration, transactional change data capture, data replication, transformations, and verification between operational and analytical enterprise systems.
- Global Data Services (GDS), new in Oracle Database 12c, provides database workload management features across replicated databases, such as Data Guard and GoldenGate.

ODP.NET supports Oracle Real Application Clusters (Oracle RAC), Data Guard, and GoldenGate transparently, meaning you do not need to change ODP.NET code to use these Oracle components. To further take advantage of these technologies, ODP.NET offers connection pooling optimization features for achieving better application high availability and performance. You can do this through configuring ODP.NET to receive, respond, and send database status messages to .NET applications.

These optimization configurations include the use of features such as Fast Application Notification (FAN), Runtime Connection Load Balancing, and Fast Connection Failover (FCF).

These connection pooling optimizations can improve high availability and performance for Oracle Real Application Clusters and Global Data Services products:

- [Fast Application Notification](#)
- [In-Band Notification](#)
- [Runtime Connection Load Balancing](#)
- [Fast Connection Failover \(FCF\)](#)

① See Also

- *Oracle Real Application Clusters Administration and Deployment Guide.*
- *Oracle Data Guard Concepts and Administration.*

Fast Application Notification

Fast Application Notification (FAN) is a high availability and load balancing notification mechanism that Oracle RAC, Data Guard, and GoldenGate use to notify ODP.NET applications about cluster configuration and service-level information, including status changes such as UP or DOWN events and server load. FAN UP and DOWN events can apply to instances, services, and nodes. Based on information received, ODP.NET can adjust its connection pool accordingly to improve application availability and performance.

With out-of-band FAN, Oracle RAC, Data Guard, and GoldenGate, use Oracle Notification Service (ONS) as its messaging infrastructure to send notifications to ODP.NET applications.

[Table 3-9](#) describes when each messaging system is used and the ODP.NET-related client configuration.

Table 3-9 Configurations for ODP.NET Driver Types

ODP.NET Driver Type	FAN Infrastructure	Configuration	Manual ONS Configuration Locations
Core and managed	ONS	Automatic or Manual	Either of these two files: • .NET configuration file • ONS configuration file
unmanaged	ONS	Automatic or Manual	oraaccess.xml file

For automatic ONS configuration, developers can add more nodes and ports for ODP.NET to listen to, in addition to the nodes and ports that ODP.NET obtains from the database automatically.

In earlier Oracle database versions, Oracle Database Advanced Queueing (AQ) provided the messaging infrastructure prior to ONS. ODP.NET applications do not require code changes to migrate from the AQ to ONS FAN infrastructure. However, some ODP.NET client configuration changes may be necessary when migrating to ONS, a newer database server version, or from ODP.NET, Unmanaged Driver to the managed driver, as documented above.

On the database server side, FAN must be set up and configured for out-of-band messages.

Using out-of-band FAN from the database, ODP.NET can do the following:

- With Runtime Connection Load Balancing, ODP.NET load balances connections among Oracle RAC nodes, services, and service members and GDS resources. This feature improves ODP.NET response time and ensures better resource allocation of server resources.
- With the Fast Connection Failover (FCF) feature, Oracle RAC, Data Guard, and GoldenGate can inform the ODP.NET connection pool if database nodes, services, service members, or the databases have gone down. These DOWN messages indicate which connections in the pool are invalid and must be removed.

① See Also

- [onsConfig section](#) to configure the .NET configuration file.
- [Client Side ONS Daemon Configuration](#) to configure the ONS configuration file.
- [Configuring a Port to Listen for Database Notifications](#) for more information on AQ configuration for FAN.
- [Runtime Connection Load Balancing](#)
- *Oracle Real Application Clusters Administration and Deployment Guide* for more information about FAN.
- *Oracle Call Interface Programmer's Guide* to configure `oraaccess.xml`.
- *Oracle Database Development Guide*

In-Band Notification

Oracle recommends using out of band notifications when possible as they support more high availability event types than in band and can be more reliable. In some scenarios, such as cloud deployments and when firewalls block notification messages between the database and client, out of band messages and using ONS may not be possible. ODP.NET can use in-band notifications instead in these scenarios. In-band notifications support the following usage scenarios:

- Pluggable database relocation
- Database service relocation
- Database rolling upgrade
- Connection Manager planned maintenance

In-band notifications rely on existing ODP.NET connections to communicate messages and for connection pooling to be enabled. ODP.NET will check for notifications every time a connection makes a database round trip, is checked in, or checked out. When a DOWN notification is received, ODP.NET will scan the pool for affected connections and close them. Applications should not see any errors as long as connections are returned to the pool prior to the drain timeout.

To use in-band notifications, no database server nor ODP.NET configuration changes are required. They are enabled automatically in ODP.NET Core and managed 19.9 and unmanaged ODP.NET 19.10 or higher versions. On the server side, in-band notifications are automatically enabled in Oracle Database 18c or higher versions.

As an HA best practice, out of band notifications should be enabled along with the default in-band notifications when possible.

Runtime Connection Load Balancing

With Runtime Connection Load Balancing, Oracle Data Provider for .NET balances work requests across Oracle RAC instances based on the load balancing advisory and service goal. Because workloads can constantly change, load balancing occurs when the application requests a new connection. Thus, ODP.NET optimizes service levels by connecting users to the least loaded nodes in real-time.

In Oracle Database 12c, Runtime Connection Load Balancing has been extended to Oracle Data Guard and Oracle GoldenGate so that ODP.NET 12c connections can be load balanced with these two database services as part of Global Data Services. No ODP.NET applications require code changes to use Global Data Services if they are already using Runtime Connection Load Balancing.

When Runtime Connection Load Balancing is enabled:

- The ODP.NET connection pool dispenses connections based on the load balancing advisory and service goal.
- The ODP.NET connection pool also balances the number of connections to each service member providing the service, based on the load balancing advisory and service goal.

By default, ODP.NET is enabled to receive Runtime Connection Load Balancing FAN messages from the server. The feature has been enabled via the "Load_Balancing=true" and "pooling=true" settings in the connection string, which are the default values. This feature can only be used if "pooling=true". In order to use Runtime Connection Load Balancing, specific Oracle server configurations must be set.

The following connection string example enables Runtime Connection Load Balancing:

```
"user id=scott;password=tiger;data source=erp;load balancing=true;"
```

📘 See Also

- [Table 3-6](#)
- ["Configuring a Port to Listen for Database Notifications"](#)
- *Oracle AI Database Net Services Administrator's Guide* to set up the Oracle Net configuration that Runtime Connection Load Balancing requires
- *Oracle Real Application Clusters Administration and Deployment Guide* for the required Oracle RAC configuration
- *Oracle AI Database Global Data Services Concepts and Administration Guide*

Fast Connection Failover (FCF)

When an Oracle RAC service, service member, node, or a Data Guard database fails, the severed ODP.NET connection objects may continue to exist in the application. If users attempt to use these invalid connections, they will encounter errors. FCF enables ODP.NET to free these severed connections proactively and quickly. Users then will be able to use the application after a server side failure without manual intervention from an administrator.

In Oracle Database 12c, FCF has been extended to Oracle Data Guard and Oracle GoldenGate for ODP.NET 12c connections through Global Data Services. No ODP.NET applications require code changes to use Global Data Services if they already use FCF.

ODP.NET applications can enable FCF through the High Availability Events, "HA Events", connection string attribute. When HA Events are enabled:

- ODP.NET connection pool proactively removes connections from the pool when a Global Data Service or Oracle RAC service, service member, node, or database goes down.

- ODP.NET proactively forces threads waiting for responses from the downed database to exit out from the existing call to avoid any hangs. When such a connection is then returned to the pool, any resource associated with that connection is freed.
- ODP.NET establishes connections to existing Oracle instances if the removal of severed connections brings the total number of connections below the "min pool size", upon the next connection request.

By default, ODP.NET is enabled to receive FCF FAN messages from the server. This feature have been enabled via the HA Events=true and pooling=true settings in the connection string, which are the default values.

The following connection string example enables HA Events:

```
"user id=scott;password=tiger;data source=erp;HA events=true;"
```

① See Also

- [Table 3-6](#)
- [Configuring a Port to Listen for Database Notifications](#)
- *Oracle AI Database Net Services Administrator's Guide* to set up the Oracle Net configuration that FCF requires
- *Oracle Real Application Clusters Administration and Deployment Guide* for the required Oracle RAC configuration
- *Oracle AI Database Global Data Services Concepts and Administration Guide*
- *Oracle Database High Availability Overview and Best Practices*

Using FCF Planned Outage to Minimize Service Disruption

FCF not only provides high availability services for unplanned outages, such as node failures, but also for planned outages, such as server repairs, upgrades, and changes, to minimize service disruption to ODP.NET application users.

When a database service is set to be stopped or relocated, a FAN message is published with a planned reason code. A FCF-aware ODP.NET connection pool (HA Events=true) receives the notification and commences to close idle connections, no longer allowing new connections to that specific database service. Active connections to that specific database service remain until users complete their tasks and the connection is returned to the pool. Thus, no users must stop work mid-stream due to a planned outage.

Eventually, all users complete their tasks and no connections remain to that database service. The database administrator can then stop the service for the planned outage task. This feature allows the database service to be stopped as quickly as possible without end user disruption.

Oracle planned outage support works with Oracle Real Application Clusters (Oracle RAC), Oracle Data Guard, and some single instance scenarios.

Oracle RAC Planned Outage

A typical planned outage scenario for Oracle RAC follows below. Note that the database server commands apply to Oracle RAC 12c Release 2 or higher. Commands for earlier releases may be different.

1. There is a need to upgrade, patch, or repair a software or hardware issue on a database server. Stop the instance gracefully such that existing users experience no to few errors. You can wait until all users complete their work before doing so. Business requirements will dictate whether you wait for all users to log out or begin the planned outage after a set time. An administrator could issue the following command line operation using Oracle Server Control Utility (srvctl):

```
srvctl relocate service -database <unique database name> -service <service name> -
drain_timeout 120 -stopoption IMMEDIATE -oldinst <existing instance>
```

This command relocates the database service from the existing instance to any instance it is configured to run on. Oracle Cluster Ready Services (CRS) will choose this instance, as the command line specifies no target. CRS will wait 120 seconds (-drain_timeout 120) for any active sessions to drain, after which any sessions remaining on the existing instance will be forcibly disconnected (-stopoption IMMEDIATE). If Application Continuity is used in conjunction with planned outage, an attempt is made to recover these killed sessions, masking the outage from end users.

The relocate operation starts the service in the new location prior to stopping the service in its existing location. Immediate relocation allows draining with no brownout. If the service cannot be started, it is not stopped at the original location to maintain availability.

2. Meanwhile in the connection pool, the FAN planned DOWN event clears idle sessions for the instance being shutdown from the ODP.NET connection pool immediately and marks that instance's active sessions to be released at the next check-in. These FAN actions drain the sessions from this instance without disrupting the users.

Existing connections on other instances remain usable, and new connections can be opened to these other instances.

3. Not all sessions will check their connections into the pool immediately. The timeout period specified by -drain_timeout after which the instance is forcibly shut down, evicting any remaining client connections. Administrators can check whether any active sessions to the instance remain by querying the v\$session table.
4. Once the upgrade, patch, or repair is complete, restart the instance and the service on the original node. The FAN UP event will inform the ODP.NET pool that it can now use the original machine again.

Oracle Data Guard Planned Outage

Oracle Data Guard performs switchovers from primary databases to standby databases in planned failover scenarios. During the switchover, administrators will want to limit end user disruptions. In Oracle Database 12c Release 2 and higher, these administrators can use the Data Guard command-line interface (DGMGRL) command to switch roles between primary and standby databases:

```
SWITCHOVER TO <database name> [WAIT <timeout in seconds> ];
```

The WAIT option specifies to wait for sessions to drain before proceeding with the switchover.

Similar to the Oracle RAC scenario, FAN informs the ODP.NET to remove idle connections from the pool. Connections subsequently checked in are destroyed until no active connections remain to that primary database, which will allow the switchover to begin.

When switchover to the standby completes, a FAN UP event informs ODP.NET that it can start creating connections to the standby instance.

During the Data Guard service relocation process, new incoming connection requests will not be accepted until the service has fully relocated. Incoming connection requests arriving during

the interim, such as in the middle of an Oracle Data Guard switchover, will receive connectivity errors.

To prevent these errors, ODP.NET can pause connection attempts until the new database service is available. ODP.NET blocks any connection attempts until the service is up or until the configured time limit expires from the time when the service DOWN event was received. This feature is useful for planned outages and service relocations. It works with Oracle RAC and Oracle Data Guard.

This time limit is the `ServiceRelocationConnectionTimeout` setting, which can be set in the `.NET` configuration file.

① **See Also**

[ServiceRelocationConnectionTimeout](#)

Pool Behavior in an Oracle RAC Database

When connection pools are created for a single-instance database, pool size attributes are applied to the single service. Similarly, when connection pools are created for an Oracle RAC database, the pool size attributes are applied to a service and not to service members. For example, if "Min Pool Size" is set to N , then ODP.NET does not create N connections for each service member. Instead, it creates, at minimum, N connections for the entire service, where N connections are distributed among the service members.

The following pool size connection string attributes are applied to a service.

- Min Pool Size
- Max Pool Size
- Incr Pool Size
- Decr Pool Size

ODP.NET connects to the same Oracle RAC node when required by a distributed transaction that has already begun on a particular node, by an Oracle runtime connection load balancing advisory, or by Oracle RAC load balancing gravitation in which connections will gravitate to an under utilized node. If the connection pool has no idle connections to this particular node, then ODP.NET will create a new connection to this node. Node affinity is honored even when the connection pool runs out of idle connections to dispense.

In the case of ODP.NET Core experiencing an unplanned database outage failing over to another database instance, connections may experience `System.PlatformNotSupportedException` errors upon an `Open()` method invocation on the `OracleConnection` object. This exception is a result of attempting to promote a local transaction to a distributed transaction due to multiple database instances having to be involved in the transaction due to the unplanned outage. The `PlatformNotSupportedException` occurs because `.NET (Core)` has limited support for distributed transactions. This behavior does not occur with managed ODP.NET nor unmanaged ODP.NET as `.NET Framework` supports distributed transactions.

Using Transaction Guard to Prevent Logical Corruption

Transaction Guard allows managed and unmanaged ODP.NET applications to use at-most-once execution in case of planned and unplanned outages and repeated submissions. Without

Transaction Guard, applications that attempt to retry operations following outages can cause logical corruption by committing duplicate transactions.

After an outage, one of the traditional problems for recovering applications had been the non-durable commit message sent back to the client. If there is a break between the client and the server, the client sees an error message indicating that the communication failed, also known as a recoverable error. This error does not inform the application if the submission executed any commit operations, or if a procedural call ran to completion while executing all expected commits. The error also does not indicate session state changes or intermittent failures. The client is left wondering if the transaction committed and if it fully completed.

These recoverable errors may require end users or applications to attempt replay by issuing duplicate transaction submissions or other forms of logical corruption. The transaction cannot be validly resubmitted if the non-transactional state is incorrect or if it is committed. Continuing to process a committed but not completed call can result in the application using a database session that is in the wrong state.

ODP.NET and Transaction Guard

Transaction Guard allows ODP.NET, Managed Driver and ODP.NET, Unmanaged Driver to eliminate duplicate transactions automatically and transparently, and in a manner that scales.

When a failure occurs, such as a node, network, or database failure, ODP.NET applications can deterministically conclude whether the transaction committed by querying its status, if the database service is up. Oracle retains the transaction status automatically, even after one of these failures.

In ODAC 12c Release 4, using Transaction Guard application development has been streamlined, reducing the application logic needed to determine the transaction outcome. Moreover, these benefits are available to both managed and unmanaged ODP.NET.

When a recoverable error is raised by a Transaction Guard enabled database service upon a database commit or upon a SQL or PL/SQL execution, which could have called a commit, then an ODP.NET `OracleException` is created with an `OracleLogicalTransaction` instance. `OracleLogicalTransaction` is always non-null. The database maintains the outcome of the logical transaction for the retention period specified by the administrator. ODP.NET automatically queries the database on behalf of the application when a recoverable error occurs so that the `OracleLogicalTransaction` object instance on the `OracleException` object can indicate whether the transaction has committed or not and whether the user call has completed or not.

If the status is committed, then the transaction has completed successfully. No other action is likely needed by the administrator.

If not committed, then ODP.NET applications can learn the current transaction state, whether it is recoverable, and whether it can be retried using `OracleLogicalTransaction`. If the error is recoverable, then the transaction is safe to re-submit. If the error is not recoverable, the application will need to determine the transaction outcome using an alternative mechanism.

Note

Transaction Guard supports only local transactions. It does not support distributed transactions.

The Transaction Guard feature is enabled or disabled through the Oracle service-level configuration through the `COMMIT_OUTCOME` setting. By default, it is not enabled. This setting can be changed without bringing down the database. Only new connections created against the service will use the new setting.

Here's an example of setting the `COMMIT_OUTCOME` using `SRVCTL`:

```
srvctl modify service -d orcl -s GOLD -commit_outcome TRUE
```

Note

Grant the `EXECUTE` privilege on the `DBMS_APP_CONT` package to the database users that retrieve the transaction status:

```
GRANT EXECUTE ON DBMS_APP_CONT TO <user name> ;
```

The following is an example ODP.NET Transaction Guard application scenario:

An ODP.NET application receives a Fast Application Notification (FAN) down event or error. FAN automatically aborts the dead session and the application receives an `OracleException`. A Transaction Guard application built to handle errors transparently would do the following:

1. `OracleException.OracleLogicalTransaction.LogicalTransactionId` property

The value returned by this property will be non-null if

- Transaction Guard is enabled,
- Transparent Application Failover (TAF) is not enabled on the connection, and
- ODP.NET was not able to determine the outcome of the transaction.

For a given `OracleException` object, if `OracleLogicalTransaction.Committed` and `OracleLogicalTransaction.UserCallCompleted` return all nulls, then the error is either a non-recoverable error, Transaction Guard is not enabled, or TAF is enabled on the connection. In any of these cases, the application should rollback then re-submit the transaction.

The `LogicalTransactionId` property will return null even if Transaction Guard is enabled and a recoverable error happens, if `OracleLogicalTransaction.Committed` is true or false.

2. `OracleException.OracleLogicalTransaction.Committed` property

This property returns true or false, depending on whether the transaction has been committed or not. If it returns null, then

- ODP.NET could not determine the outcome of the transaction,
- Error is not recoverable,
- Transaction Guard was not enabled, or
- TAF is enabled on the connection.

3. `OracleException.OracleLogicalTransaction.UserCallCompleted` property

This property returns true if the user call has completed, else it returns false. This property returns null if

- Transaction Guard was not enabled,
- Error is not recoverable,
- TAF is enabled on the connection, or

- ODP.NET is not able to determine if the user call has completed or not.

See the following table for the implications of what Committed and UserCallCompleted values mean.

Table 3-10 Implication of Committed and UserCallCompleted Values

Committed Value	UserCallCompleted Value	Outcome
True	True	The transaction was successful. The result can be returned to the application.
False	False	The transaction was not successful. The application can resubmit the transaction again.
True	False	The transaction committed, but there may be additional state, such as row counts or nested PL/SQL logic, that prevents the application from continuing as expected.

Example 3-4 Using Transaction Guard: Sample Code

```
using System;
using Oracle.DataAccess.Client;
//alternatively can use using Oracle.ManagedDataAccess.Client;

class TransactionGuardSample
{
    static void Main()
    {
        bool bReadyToCommit = false;

        string constr = "user id=hr;password=hr;data source=oracle";
        OracleConnection con = new OracleConnection(constr);
        OracleTransaction txn = null;
        OracleCommand cmd = null;

        try
        {
            string sql = " update employees set salary=10000 where
employee_id=103";
            con.Open();
            txn = con.BeginTransaction();
            cmd = new OracleCommand(con, sql);
            cmd.ExecuteNonQuery();
            bReadyToCommit = true;
        }
        catch (Exception ex)
        {
            // rollback here as the SQL execution is unsuccessful
            txn.Rollback();
            Console.WriteLine(ex.ToString());
        }
    }
}
```



```

        try
        {
            if (bReadyToCommit)
                txn.Commit();
        }
        catch (Exception ex)
        {
            if (ex is OracleException)
            {
                // It's safe to re-submit the work if the error is
                // recoverable and the transaction has not been committed
                if (ex.IsRecoverable && ex.OracleLogicalTransaction.Committed
                    == false)
                {
                    // safe to re-submit work
                }
                else
                {
                    // do not re-submit work
                }
            }
        }
        finally
        {
            // dispose all objects
            txn.Dispose();
            cmd.Dispose();
            con.Dispose(); // place the connection back to the connection pool
        }
    }
}

```

Transaction Guard is not supported when Transparent Application Failover (TAF) is enabled for the service. As such, when TAF is enabled, the `LogicalTransactionId` will always be null and the transaction outcome will not be determined.

① See Also

- ["OracleLogicalTransaction Class"](#)
- *Oracle Database Development Guide* for more information on Transaction Guard

Application Continuity

Oracle Application Continuity enables database requests to automatically replay transactional or non-transactional operations in a non-disruptive and rapid manner in the event of a severed database session, which results in a recoverable error. Application Continuity improves end-user experience by masking planned and unplanned related errors. Applications can be developed without complex logic to handle exceptions, while automatically replaying database operations upon a recoverable error.

Without Application Continuity, it is almost impossible to mask outages in a safe and reliable manner. Common issues encountered include:

- The client state remains at present time, with entered data, returned data, and variables cached, while the database state changes are lost.
- If a transaction commit has occurred, the commit message is not durable. Moreover, checking a lost request does not guarantee that it will not commit after being checked.
- Non-transactional database session state is lost.
- If the request can continue, the database and the client session must be synchronized.

Application Continuity is a feature available with the Oracle Real Application Clusters (RAC), Oracle RAC One Node, Oracle Active Data Guard, and Oracle Autonomous Database in both Shared and Dedicated Infrastructure.

In Oracle Database 18c, Application Continuity improves transparent session and transactional state tracking and recording of the database session to enable recovery following recoverable outages. This enhancement is called Transparent Application Continuity (TAC). TAC has no reliance on application knowledge or application code changes, allowing it to be enabled for your applications. ODP.NET application transparency and failover are achieved by consuming the state-tracking information that captures and categorizes the session state usage as the application issues user calls.

ODP.NET and Application Continuity

All ODP.NET provider types, core, managed, and unmanaged, support Application Continuity and Transparent Application Continuity.

Note

Asynchronous ODP.NET does not support Application Continuity and Transparent Application Continuity.

With Application Continuity or TAC enabled, ODP.NET ensures all the application's executed statements are logged appropriately so that they can be replayed upon a recoverable error. This applies for all application SQL and PL/SQL, as well as any internal ODP.NET operations.

On the client side, Application Continuity or TAC is enabled by setting the ODP.NET connection string attribute, `Application Continuity=true`. The attribute is set to true by default.

If Application Continuity is set to true, but the database server does not enable Application Continuity or TAC, then ODP.NET will still create new connections. However, these connections will not be Application Continuity enabled.

Both Application Continuity and TAC can be used with ODP.NET connection pool by implicitly `BeginRequest/EndRequest` calls, as well as without ODP.NET connection pool, by explicitly calling `BeginRequest/EndRequest` calls.

① See Also

- *Oracle Call Interface Developer's Guide*
- *Oracle Real Application Clusters Administration and Deployment Guide*
- *Oracle Database High Availability Overview and Best Practices* for more information about High Availability Overview and Best Practices - Configuring Level 3: Mask Unplanned and Planned Failovers from Applications.

Application Continuity Work Requests

AC and TAC operate on request boundaries. A request is an application unit of work. Typically, it exists between when an application borrows and returns a database connection from the connection pool. All ODP.NET providers use this model by default to demarcate work request boundaries.

ODP.NET Core and managed ODP.NET include an additional capability for developers to explicitly identify request boundaries themselves using the `OracleConnection.BeginRequest` and `EndRequest` methods. Preferably, applications will not need to use these methods and just rely on the default behavior. When a connection is checked out, ODP.NET implicitly calls `BeginRequest`. And when the connection is checked in, ODP.NET implicitly calls `EndRequest`. This model scales well and makes AC and TAC easier for developer use.

Applications that do not use ODP.NET connection pools should explicitly mark request boundaries, such as the case with custom connection pools. The `BeginRequest` method should be called upon connection check out and the `EndRequest` method should be called upon connection check in.

With an understanding of the request demarcation points, ODP.NET knows when it is safe to release the database call history. Proper request boundary identification conserves memory consumption for long living connections, such as when they exist in a custom pool.

These methods have no application impact other than improving resource consumption, recovery, and load balancing performance. They do not alter connection state by calling any method, SQL, nor PL/SQL.

If a begin or end request is made while a local transaction is open, ODP.NET returns an error.

Transparent Application Failover

Oracle Transparent Application Failover (TAF) is a client-side high availability feature. It enables a client to automatically reconnect to a secondary database instance if the connected primary instance fails or shuts down.

No new failover code is required to use TAF. As the name implies, the feature is transparent, meaning ODP.NET and Oracle database will manage the instance failure detection and connection re-establishment process if TAF is enabled and configured.

ODP.NET Core, managed, and unmanaged provider types all support TAF. ODP.NET Core and managed started support with ODP.NET 23.3.3. There are differences in the TAF features each provider type supports.

TAF can be configured in the database or client side. On the client side, TAF settings can be made in the Oracle connect descriptor or through ODP.NET APIs.

TAF automatically restores some or all the elements associated with active database connections. If other elements require recovery, they should be added in the application code, such as within an ODP.NET TAF callback. Here are more details about ODP.NET TAF recovery features:

- Database connections

TAF automatically reestablishes the ODP.NET connection using the same connection string or an alternate connection string specified for failover.

- User database sessions

TAF automatically logs a user in with the same login credentials as originally used. If multiple users use the same connection, then TAF automatically logs them in as they attempt to process database commands. Unfortunately, TAF cannot automatically restore other session properties, but those properties can be restored by invoking a callback function.

- Completed commands

If a command completed at the time of connection failure and changed the database state, then TAF does not resend the command upon reconnection. If TAF reconnects and another command may have changed the database, then TAF issues an error message to the application. This TAF feature is available from unmanaged ODP.NET, but not managed nor core drivers.

- Open cursors for results fetching

TAF allows applications that began fetching rows from a cursor before failover to continue fetching rows after recovery. This is called select failover. It re-runs a SELECT statement using the same snapshot, discarding those rows already fetched and retrieving those rows that were not fetched initially. TAF verifies that the discarded rows are those that were returned initially, or it returns an error message. This TAF feature is available from unmanaged ODP.NET, but not managed nor core drivers.

- Active transactions

Any active transactions are rolled back at the time of failure. TAF cannot preserve active transactions after failover. The application instead receives an error message until a ROLLBACK command is submitted.

- Server-side program variables

Server-side program variables, such as PL/SQL package states, are lost during failures. TAF cannot recover them. They can be re-initialized by making a call from the failover callback.

TAF can be configured on the client side through the `FAILOVER_MODE` attribute in the TNS connect descriptor.

Table 3-11 ODP.NET TAF Failover Mode Support

TAF Failover Mode	ODP.NET Core Support	Managed ODP.NET Support	Unmanaged ODP.NET Support	Description
Session Failover	Y	Y	Y	Recreates lost connections and sessions
Select Failover	N	N	Y	Replays in-progress queries

Table 3-12 ODP.NET TAF Failover Method Support

TAF Failover Method	ODP.NET Core Support	Managed ODP.NET Support	Unmanaged ODP.NET Support	Description
Basic	Y	Y	Y	Establishes connections at failover time. This option requires almost no work on the backup database server until failover.
Preconnect	N	N	Y	Pre-establishes connections on backup instance. This provides faster failover but requires the backup instance to support all connections from the primary instance.

Note

Managed ODP.NET and ODP.NET Core do not support BACKUP and TRANSACTION TAF parameters. BACKUP specifies the failover node. TRANSACTION allows the database to complete the current transaction following a recoverable error.

TAF Notification

Given the delays that failovers can cause, applications may wish to be notified by a TAF callback. ODP.NET supports the TAF callback function through the Failover event of the OracleConnection object, which allows applications to be notified whenever a failover occurs. To receive TAF callbacks, an event handler function must be registered with the Failover event.

When Failover Occurs

When a failover occurs, the Failover event is raised and the registered event handler is invoked several times during the course of reestablishing the connection to another Oracle instance.

The first call to the event handler occurs when Oracle Database first detects an instance connection loss. This allows the application to act accordingly for the upcoming delay for the failover.

If the failover is successful, the Failover event is raised again when the connection is reestablished and usable. At this time, the application can resynchronize the OracleGlobalization session setting and inform the application user that a failover has occurred. No significant database operation should occur immediately after a FailoverEvent.Begin event. SQL and major database operations should wait until the

`FailoverEvent.End` event. `FailoverEvent.Begin` is primarily used to reject failover or to trace it. `FailoverEvent.Begin` can also be used for non-database application operations, such as informing the end user a failover is in progress and to wait until it completes before proceeding. Transactions can be used in the `FailoverEvent.End` callback phase, such as to file fault tickets or audit. These transactions must be committed before the callback completes.

If failover is unsuccessful, the Failover event is raised to inform the application that a failover did not take place.

The application can determine whether or not the failover is successful by checking the `OracleFailoverEventArgs` object that is passed to the event handler.

Registering an Event Handler for Failover

The following example registers an event handler method called `OnFailover`:

```
// C#

using System;
using Oracle.DataAccess.Client;

class TAFCallbackSample
{
    public static FailoverReturnCode OnFailover(object sender,
                                                OracleFailoverEventArgs eventArgs)
    {
        switch (eventArgs.FailoverEvent)
        {
            case FailoverEvent.Begin :
                Console.WriteLine(
                    " \nFailover Begin - Failing Over ... Please standby \n");
                Console.WriteLine(
                    " Failover type was found to be " + eventArgs.FailoverType);
                break;

            case FailoverEvent.Abort :
                Console.WriteLine(" Failover aborted. Failover will not take place.\n");
                break;

            case FailoverEvent.End :
                Console.WriteLine(" Failover ended ...resuming services\n");
                break;

            case FailoverEvent.Reauth :
                Console.WriteLine(" Failed over user. Resuming services\n");
                break;

            case FailoverEvent.Error :
                Console.WriteLine(" Failover error gotten. Sleeping...\n");
                return FailoverReturnCode.Retry;

            default :
                Console.WriteLine("Bad Failover Event: %d.\n", eventArgs.FailoverEvent);
                break;
        }
        return FailoverReturnCode.Success;
    } /* OnFailover */

    static void Main()
    {
        OracleConnection con = new OracleConnection();
```

```
con.ConnectionString = "User Id=scott;Password=tiger;Data Source=oracle;";
con.Open();
con.Failover += new OracleFailoverEventHandler(OnFailover);
Console.WriteLine("Event Handler is successfully registered");

// Close and Dispose OracleConnection object
con.Close();
con.Dispose();
}
}
```

The Failover event invokes only one event handler. If multiple Failover event handlers are registered with the Failover event, only the event handler registered last is invoked.

① Note

Distributed transactions are not supported in an environment where failover is enabled.

① See Also

- *Oracle AI Database Net Services Administrator's Guide*
- ["OracleFailoverEventHandler Delegate"](#)
- ["OracleFailoverEventArgs Class"](#)

Oracle Globally Distributed Database

Oracle Globally Distributed Database is a data tier architecture, where data is horizontally partitioned across independent databases. Each database in such a configuration is called a shard. All shards together make up a single logical database, which is referred to as a sharded database. Oracle Globally Distributed Database is a shared-nothing database architecture. The independent physical databases do not share CPU, memory, or storage devices. However, from the perspective of an application, the collection of physical databases looks like a single logical database.

Oracle Globally Distributed Database uses Global Data Services (GDS), where GDS routes a client request to an appropriate database based on parameters such as availability, load, network latency, and replication lag. A GDS pool is a set of replicated databases that offers the same global service. The databases in a GDS pool can be located in multiple data centers across different regions. A sharded GDS pool contains all shards of a sharded database and their replicas, and appears as a single sharded database to database clients.

Applications can connect to multiple databases (shards) where data is partitioned based on one or more sharding strategies. The strategy can be hash based, range based, or list based. Each time a database operation is required, the application needs to determine which shard it must connect to.

A sharding key provides the partitioning key that determines in which shard a row of data is stored. A table can be partitioned using a sharding key.

A super sharding key is a collection of shard chunks, where only those chunks, which have a specific value of the super shard key identifier, are stored. A super sharding key is used for distributing data across database groups. Specifying super sharding keys are a way through which user-controlled data partitioning is possible.

ODP.NET Sharding

Starting from version 12.2, unmanaged ODP.NET and Oracle Database both support sharding. Managed ODP.NET and ODP.NET Core started supporting sharding with version 21. All three providers support the same sharding features with the exception that unmanaged ODP.NET does not support the chunk migration connection timeout property.

Starting with ODP.NET 26ai, Raft replication and Oracle Data Guard are supported to improve failover reliability and performance. Existing apps can use both these features without any ODP.NET code changes necessary.

ODP.NET applications must provide the sharding key and super sharding key information before opening the database connection for single shard queries. These sharding values cannot be set or changed after opening the connection. If any of the shard key values need to be modified, a new connection must be created with the new values and then opened.

If shard keys are set after the connection has been opened, the ODP.NET connection will not use these new shard key values until after the next `OracleConnection.Open()` call.

The `OracleShardingKey` object stores one or more key values. Multiple keys can be set to create a composite key. ODP.NET recognizes the sharding key(s) specified and connects to the correct shard and chunk.

Sharding is supported with or without connection pooling. The ODP.NET connection pool maintains connections to different shards and chunks of the sharded GDS database within the same shared pool.

The shard key (`SHARD_KEY`) and super sharding key (`GROUP_KEY`) can be specified in the TNS connect descriptor, rather than in the application code. The .NET developer then chooses the connect descriptor applicable to the shard that the application will use.

The data distribution across the shards and chunks in the database is transparent to the end user. ODP.NET minimizes the end user impact of chunk resharding within GDS.

To perform cross-shard queries, no ODP.NET shard APIs are used. Instead, applications connect to the GDS catalog service, allowing access to all the sharded databases. The SQL query is specifically constructed to iterate over all the necessary shards. For example, the non-shard database query `select count(*) from employees` is equivalent to the cross-shard query `select sum(c) from (Iterator(select count(*) c from employees(i)))`.

ODP.NET Single Shard Query Example

```
using System;
using Oracle.DataAccess.Client;

class Sharding
{
    static void Main()
    {
        OracleConnection con = new OracleConnection("user id=hr;password=hr;Data
Source=orcl;");
        //Setting a shard key
        OracleShardingKey shardingKey = new OracleShardingKey(OracleDbType.Int32, 123);
        //Setting a second shard key value for a composite key
        shardingKey.SetShardingKey(OracleDbType.Varchar2, "gold");
    }
}
```

```

//Creating and setting the super shard key
OracleShardingKey superShardingKey = new OracleShardingKey();
superShardingKey.SetShardingKey(OracleDbType.Int32, 1000);

//Setting super sharding key and sharding key on the connection
con.SetShardingKey(shardingKey, superShardingKey);
con.Open();

//perform SQL query
}
}

```

By default, ODP.NET only dispenses connections to shards with chunks with read and write privileges available. If some chunks in a shard are read-only, ODP.NET will not dispense a connection to that shard unless `READONLY_CHUNK_OK` is set to `TRUE` in the `CONNECT_DATA` section of the connect descriptor. Connect descriptors are generally stored in `tnsnames.ora` files. Enabling `READONLY_CHUNK_OK` just means that both read/write and read-only connections can be dispensed. It does not specify only read-only connections are dispensed.

Shards are made read-only when chunks are migrating from one shard to another, usually to re-balance data or workload across shards.

Here's an example using `READONLY_CHUNK_OK` in a connect descriptor:

```

SHARDDDB =
(DESCRIPTION =
  (ADDRESS = (PROTOCOL = tcp)(HOST = myhost)(PORT = 1521))
  (CONNECT_DATA =
    (SERVICE_NAME = myservicename)
    (READONLY_CHUNK_OK=true)
  )
)

```

In ODP.NET, super shard keys can use range sharding or list sharding. With composite sharding, the shard keys must use hash sharding. The table below describes operations supported for different shard key types, not super shard keys.

Table 3-13 Supported Operations for Shard Key Types

Chunk Event Type	Hash	Range	List
Chunk Read Only <i>Not supported in unmanaged ODP.NET</i>	Yes	Yes	Yes
Chunk Up	Yes	Yes	Yes
Chunk Down	Yes	Yes	Yes
Chunk Split	Yes	Yes	Yes
Add Chunk	No	Yes	Yes
Add Chunk Values	No	No	Yes
Drop Chunk	No	Yes	Yes
Drop Chunk Values	No	No	Yes
Merge Chunk	No	Yes	Yes
Invalidate Chunk	No	Yes	Yes

Table 3-13 (Cont.) Supported Operations for Shard Key Types

Chunk Event Type	Hash	Range	List
Partitionset Split <i>Not supported in unmanaged ODP.NET</i>	Yes	No	No
New Partitionset <i>Not supported in unmanaged ODP.NET</i>	Yes	No	No

Sharding Split Partitionset

Over time, changing business needs require modifying data organization to optimize resource usage. Multiple super sharding keys grouped in the same shardspace may have been optimal under past conditions. If it becomes necessary to move one or more of the super sharding keys into their own shardspace, such as on a faster machine, then split partitionset operations can facilitate efficient data movement with little down time.

Because each chunk contains data for multiple super sharding keys, a split operation occurs first. Then, the data is moved. The operation is performed one chunk at a time. When the chunk needs to be split, its data will be read-only for a short time. The chunk's sharding key range does not change.

To limit connectivity errors while the chunk is unavailable during a chunk migration or split partitionset, developers can set the `ChunkMigrationConnectionTimeout` property to wait for the set time or whenever the operation completes, whichever comes first. Users then do not receive connection timeout errors while the migration or split occurs if the operation completes in the expected time frame.

A key benefit for developers is they do not have to maintain and track split partitionset activity in their applications. ODP.NET performs this activity for them automatically and routes connections appropriately. It has a shard routing cache to manage the sharding topology and record changes. The ODP.NET connection pool will dispense the appropriate connection for a given sharding key and super sharding key, including after a split partitionset.

ODP.NET split partitionset is supported for hash sharding only with either list or range super shard keys. Oracle Database 23.8 and higher supports sharding split partitionset.

See Also

- [OracleShardingKey Class](#)
- [SetShardingKey\(OracleShardingKey, OracleShardingKey\)](#)
- [ChunkMigrationConnectionTimeout](#)
- [ChunkMigrationConnectionTimeout](#)
- Oracle Globally Distributed Database*

OracleCommand Object

The `OracleCommand` object represents SQL statements executed on Oracle Database.

Note

Optimizer hint syntax in the form `--+ ...` is not supported. ODP.NET supports this syntax: `/*+ ... */`.

This section includes the following topics:

- [Command Auto-Commit](#)
- [Transactions](#)
- [System.Transactions and Promotable Transactions](#)
- [Distributed Transactions](#)
- [Sessionless Transactions](#)
- [Parameter Binding](#)
- [Batch Processing](#)
- [Statement Caching](#)
- [Self-Tuning](#)
- [Data Transmission Performance](#)
- [Securing Identifiers and Literals from SQL Injection Attacks](#)

Command Auto-Commit

By default, ODP.NET commits each SQL operation immediately upon execution outside of a transaction. Starting with 26ai, managed ODP.NET and ODP.NET Core can disable auto-commit so that SQL statements are executed only after an explicit commit statement.

Unmanaged ODP.NET cannot disable auto-commit without a transaction.

Developers disable auto-commit by setting the `OracleConnection AutoCommit` property to false. ODP.NET can then start transactions implicitly. After the `AutoCommit` property is set, that setting is honored for all subsequent `Open()` calls on that specific `OracleConnection` object until the `AutoCommit` value is modified.

`OracleConnection` includes APIs to manage local transactions, such as the ability to commit, rollback, and create savepoints. It can check if the application is currently within a transaction. It can set the transaction isolation level.

When `AutoCommit` is set while enlisted in an explicit (that is, `BeginTransaction`), local, or distributed (that is, `System.Transactions`) transaction, an error will occur. Calling any `OracleConnection` transaction management API while enlisted in `System.Transactions` will also result in an error.

Transactions

Oracle Database starts a transaction only in the context of a connection. Once a transaction starts, all the successive command execution on that connection run in the context of that transaction. Transactions can be started only on an `OracleConnection` object, and the read-only `Transaction` property on the `OracleCommand` object is implicitly set by the `OracleConnection` object. Therefore, the application cannot set the `Transaction` property, nor does it need to.

Explicit transactions are required with SQL statements containing "FOR UPDATE" and "RETURNING" clauses. This is not necessary if global transactions are used.

System.Transactions and Promotable Transactions

ODP.NET supports System.Transactions. A local transaction is created for the first connection opened in the System.Transactions scope to Oracle Database. When a second connection is opened, this transaction is automatically promoted to a distributed transaction. This functionality provides enhanced performance and scalability.

Connections created within a transaction context, such as TransactionScope or ServicedComponent, can be established to different versions of Oracle Database. However, in order to enable the local transaction to be promotable, the following must be true:

- The first connection in the transaction context must be established to an Oracle Database.
- Promoting local transactions requires Oracle Services for Microsoft Transaction Server. If this requirement is not met, then a second connection request in the same transaction context throws an exception.

Transaction promotion will throw an ORA-24797 error when the database transaction is already distributed due to the use of database links.

If applications use System.Transactions, it is required that the enlist connection string attribute is set to either true (default) or dynamic. However, enlist=dynamic cannot be used with TransactionScope because auto-enlistment requires enlist=true.

ODP.NET supports the following System.Transactions programming models for applications using distributed transactions.

- [Implicit Transaction Enlistment Using TransactionScope](#)
- [Explicit Transaction Enlistment Using CommittableTransaction](#).

Implicit Transaction Enlistment Using TransactionScope

The TransactionScope class provides a mechanism to write transactional applications where the applications do not need to explicitly enlist in transactions. To accomplish this, the application uses the TransactionScope object to define the transactional code. Connections created within this transactional scope will enlist in a local transaction that can be promoted to a distributed transaction.

Note that the application must call the Complete method on the TransactionScope object to commit the changes. Otherwise, the transaction is aborted by default.

```
// C#

using System;
using Oracle.DataAccess.Client;
using System.Data;
using System.Data.Common;
using System.Transactions;

class psfTxnScope
{
    static void Main()
    {
        int retVal = 0;
        string providerName = "Oracle.DataAccess.Client";
        string constr =
```

```

        @"User Id=scott;Password=tiger;Data Source=oracle;enlist=true";

// Get the provider factory.
DbProviderFactory factory = DbProviderFactories.GetFactory(providerName);

try
{
    // Create a TransactionScope object, (It will start an ambient
    // transaction automatically).
    using (TransactionScope scope = new TransactionScope())
    {
        // Create first connection object.
        using (DbConnection conn1 = factory.CreateConnection())
        {
            // Set connection string and open the connection. this connection
            // will be automatically enlisted in a promotable local transaction.
            conn1.ConnectionString = constr;
            conn1.Open();

            // Create a command to execute the sql statement.
            DbCommand cmd1 = factory.CreateCommand();
            cmd1.Connection = conn1;
            cmd1.CommandText = @"insert into emp (empno, ename, job) values
                                (1234, 'emp1', 'dev1')";

            // Execute the SQL statement to insert one row in DB.
            retVal = cmd1.ExecuteNonQuery();
            Console.WriteLine("Rows to be affected by cmd1: {0}", retVal);

            // Close the connection and dispose the command object.
            conn1.Close();
            conn1.Dispose();
            cmd1.Dispose();
        }

        // The Complete method commits the transaction. If an exception has
        // been thrown or Complete is not called then the transaction is
        // rolled back.
        scope.Complete();
    }
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
    Console.WriteLine(ex.StackTrace);
}
}
}

```

Explicit Transaction Enlistment Using CommittableTransaction

The instantiation of the `CommittableTransaction` object and the `EnlistTransaction` method provides an explicit way to create and enlist in a transaction. Note that the application must call `Commit` or `Rollback` on the `CommittableTransaction` object.

```

// C#

using System;
using Oracle.DataAccess.Client;
using System.Data;
using System.Data.Common;
using System.Transactions;

```

```
class psfEnlistTransaction
{
    static void Main()
    {
        int retVal = 0;
        string providerName = "Oracle.DataAccess.Client";
        string constr =
            @"User Id=scott;Password=tiger;Data Source=oracle;enlist=dynamic";

        // Get the provider factory.
        DbProviderFactory factory = DbProviderFactories.GetFactory(providerName);

        try
        {
            // Create a committable transaction object.
            CommittableTransaction cmtTx = new CommittableTransaction();

            // Open a connection to the DB.
            DbConnection conn1 = factory.CreateConnection();
            conn1.ConnectionString = constr;
            conn1.Open();

            // enlist the connection with the committable transaction.
            conn1.EnlistTransaction(cmtTx);

            // Create a command to execute the sql statement.
            DbCommand cmd1 = factory.CreateCommand();
            cmd1.Connection = conn1;
            cmd1.CommandText = @"insert into emp (empno, ename, job) values
                                (1234, 'emp1', 'dev1')";

            // Execute the SQL statement to insert one row in DB.
            retVal = cmd1.ExecuteNonQuery();
            Console.WriteLine("Rows to be affected by cmd1: {0}", retVal);

            // commit/rollback the transaction.
            cmtTx.Commit(); // commits the txn.
            //cmtTx.Rollback(); // rolls back the txn.

            // close and dispose the connection
            conn1.Close();
            conn1.Dispose();
            cmd1.Dispose();
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
            Console.WriteLine(ex.StackTrace);
        }
    }
}
```

See Also

["EnlistTransaction"](#)

Distributed Transactions

A distributed transaction includes one or more statements that, individually or as a group, update data on two or more distinct nodes of a distributed database.

ODP.NET Core, Managed, and Unmanaged Drivers integrate with Microsoft Distributed Transaction Coordinator (MSDTC) and Oracle databases. MSDTC coordinates with all the resource managers that are enlisted to the same System.Transactions object, to perform a 2-phase commit or rollback atomically. With that, Oracle distributed transactions can then be committed or rolled back across networks properly.

ODP.NET only supports the read committed isolation level for distributed transactions.

ODP.NET Core 23.9 and higher supports distributed transactions on Windows operating systems only. .NET Core applications require [TransactionManager.ImplicitDistributedTransactions](#) to be set to true to use distributed transactions. Failure to do so will result in an ORA-50130.

Microsoft Distributed Transaction Coordinator Integration

Managed ODP.NET and ODP.NET Core include a native fully managed implementation that supports integration with MSDTC for distributed transactions enlistments, commits, and rollbacks.

For unmanaged ODP.NET, Oracle Services for Microsoft Transaction (OraMTS) provides MSDTC integration that allows client components to participate in Oracle database distributed transactions. OraMTS act as a proxy for the Oracle database to MSDTC to ensure that Oracle distributed database transactions commit or rollback together with the rest of the distributed transaction.

Unmanaged ODP.NET can use either a managed or unmanaged OraMTS implementation. Oracle recommends using the managed OraMTS for applications requiring high availability from Oracle RAC or Data Guard.

If a failure occurs in a distributed transaction, such as a network failure or server hardware failure, then it can leave an in-process transaction in-doubt. All ODP.NET provider types use the OraMTS recovery service to resolve these in-doubt transactions. This recovery service runs as a Windows service. Administrators must install and configure the OraMTS Windows recovery service to manage recovery scenarios whether they use managed or unmanaged ODP.NET. Only one recovery service is needed per machine.

It is required to install the OraMTS Recovery Service on all the client machines where ODP.NET is running and participating in MSDTC. As a machine may have multiple IP addresses, administrators for managed ODP.NET applications can specify the host machine name or IP address that has the running recovery service in the application's .NET configuration file. ODP.NET, Unmanaged Driver resolves the IP/machine name for the recovery service automatically.

Table 3-14 ODP.NET Distributed Transaction Support Matrix

ODP.NET Type	Minimum Version	Distributed Transaction Support
Core	.NET 8	NET's distributed transaction implementation
Managed	.NET Framework 4.8	.NET Framework's native managed implementation for distributed transactions.

Table 3-14 (Cont.) ODP.NET Distributed Transaction Support Matrix

ODP.NET Type	Minimum Version	Distributed Transaction Support
Unmanaged	.NET Framework 4.8	OraMTS (default) or managed OraMTS implementation. Oracle recommends using managed OraMTS for unmanaged ODP.NET applications requiring high availability from Oracle RAC or Data Guard.

① See Also

Manually Creating an Oracle MTS Recovery Service in *Oracle Services for Microsoft Transaction Server Developer's Guide for Microsoft Windows*.

ODP.NET Core and Managed Driver Setup

This section explains the setup and configuration steps required for using distributed transactions with ODP.NET Core and Managed Driver. Follow these steps to configure distributed transactions:

1. Create and setup the OraMTS recovery service or make sure an existing recovery service is running.
2. Set the port number the OraMTS recovery service listens on to resolve in-doubt transactions. The property can be set on `OracleConfiguration`, `OracleDataSourceBuilder`, or .NET configuration file.

ODP.NET, Unmanaged Driver Setup

① Note

ODP.NET, Unmanaged Driver is deprecated in 26ai. See Oracle Database Upgrade Guide for more information.

This section explains the setup and configuration steps required for using distributed transactions with ODP.NET, Unmanaged Driver.

ODP.NET, Unmanaged Driver includes managed OraMTS in its assembly. OraMTS is the default option for ODP.NET, Unmanaged Driver, to ensure implementation continuity during upgrades. However, Oracle recommends the managed OraMTS option when using any high availability FAN operations (HA Events = true) with Oracle Real Application Clusters or Oracle Data Guard. The managed OraMTS option support high availability functionality, while the traditional OraMTS does not.

Applications can set OraMTS (default) or managed OraMTS usage through the `UseOraMTSManaged` parameter in the .NET configuration file.

Install and configure OraMTS, including its recovery service to use traditional OraMTS implementation for ODP.NET, Unmanaged Driver.

To set managed OraMTS up, perform the following steps:

1. Set UseOraMTSManaged to true in the .NET configuration file.
2. Create and setup the OraMTS recovery service or make sure an existing recovery service is running.

① See Also

distributedTransaction section for more information about .NET configuration setup

Sessionless Transactions

A transaction that can perform work and span across multiple database sessions is called a sessionless transaction.

Oracle database sessionless transactions enable apps to start a transaction, submit a unit of work, and suspend the transaction in one session. The transaction can then be resumed in a different database session, any session, and continue to perform work before committing or rolling back.

A local transaction must operate within the same session, which is a limitation sessionless transactions do not have. No transaction manager, such as distributed transaction coordinator (DTC), is required for sessionless transactions that span across database sessions. Transaction manager involvement generally requires higher overhead and slows performance to perform two-phase commit and might result in in-doubt transaction when network failure occurs.

With sessionless transactions, a transaction manager, like DTC, is not needed. The database internally coordinates the two-phase commit (2PC) protocol without the need for any application-side logic. Therefore, there is no risk of in-doubt transactions and no need for any recovery mechanism.

With sessionless transactions, neither keeping a connection open for long periods of time to support a long running transaction nor a transaction manager is necessary. This flexibility gives transactional apps more possibilities to make design choices that are efficient, easier to manage, and better performing.

Oracle Database release 23.6 introduced support for sessionless transactions on the server side. Managed ODP.NET and ODP.NET Core 23.7 established client side sessionless transaction support.

ODP.NET sessionless transactions do not support promotable nor distributed transactions. They do not support transaction savepoints nor Transparent Application Failover. They also do not support temporary tables as temporary table data is not allowed to span sessions.

Sessionless transaction support with Application Continuity and Transparent Application Continuity will be enabled in a future ODP.NET release.

Sessionless Transaction Operations

An ODP.NET sessionless transaction can have the following operations:

- **Begin** – Start a sessionless transaction in an ODP.NET session using `OracleConnection.BeginSessionlessTransaction()`.
- **Suspend** – Suspend a sessionless transaction to allow it to be resumed later in a different session.

- Resume – Resume the transaction in a different or the same session. Further DML can occur until the transaction finally commits or aborts.
- Commit – Commit the sessionless transaction and permanently save changes.
- Abort – Roll back the sessionless transaction without saving the transaction changes.

Sessionless Transaction ODP.NET Code Sample

```
OracleConnection conn1 = new OracleConnection(ConnStr1);
conn1.Open();

//Specify sessionless transaction options.
//Parameter 1 (timeout): Specify transaction timeout, such as 100 seconds
//Parameter 2 (start behavior): Specify the transaction start behavior, such as start
before the command execution
//Parameter 3 (resume behavior): Specify the resume transaction behavior, such as resume
before the command execution
//Parameter 4 (suspend sessionless transaction after execution): Specify whether to
suspend the transaction after command execution, such as true to suspend
conn1.SessionlessTransactionOptions = new OracleSessionlessTransactionOptions(100,
OracleSessionlessTransactionStartBehavior.StartBeforeExecution,
OracleSessionlessTransactionStartBehavior.StartBeforeExecution, true);

//Begin sessionless transaction. The transaction ID can be used to resume in separate
session when the transaction is suspended.
byte[] txnId = conn1.BeginSessionlessTransaction();

//Execute data update in first connection.
//The sessionless transaction is started, the insert is executed under the transaction
context and the sessionless transaction is suspended after the execution.
OracleCommand cmd = new OracleCommand("insert into emp values (23, 'abc')", conn1);
cmd.ExecuteNonQuery();

//Open new connection to same database
OracleConnection conn2 = new OracleConnection(connStr);
conn2.Open();

//Resume the sessionless transaction using the transaction ID
conn2.ResumeSessionlessTransaction(txnId);

//Execute a data update with the new connection in the existing sessionless transaction
OracleCommand cmd2 = new OracleCommand("insert into dept values (2, 'dept1')", conn2);

//The sessionless transaction resumes and the second DML is executed under the same
sessionless transaction context.
cmd2.ExecuteNonQuery();

//Execute another command, the third DML is executed under the same sessionless
transaction context
cmd2.CommandText = "insert into emp values (25, 'def')";
cmd2.ExecuteNonQuery();

//Commit the sessionless transaction. All three DMLs commit.
conn2.Commit();
```

See Also

Developing Applications with Sessionless Transactions

Parameter Binding

When the `DbType` property of an `OracleParameter` object is set, the `OracleDbType` property of the `OracleParameter` object changes accordingly, or vice versa. The parameter set last prevails. An application can bind the data and have ODP.NET infer both the `DbType` and `OracleDbType` properties from the .NET type of the parameter value. ODP.NET allows applications to obtain an output parameter as either a .NET Framework type or an ODP.NET type. The application can specify which type to return for an output parameter by setting the `DbType` property of the output parameter (.NET type) or the `OracleDbType` property (ODP.NET type) of the `OracleParameter` object. For example, if the output parameter is set as a `DbType.String` type by setting the `DbType` property, the output data is returned as a .NET `String` type. On the other hand, if the parameter is set as an `OracleDbType.Char` type by setting the `OracleDbType` property, the output data is returned as an `OracleString` type. If both `DbType` and `OracleDbType` properties are set before the command execution, the last setting takes affect.

ODP.NET populates `InputOutput`, `Output`, and `ReturnValue` parameters with the Oracle data, through the execution of the following `OracleCommand` methods:

- `ExecuteReader`
- `ExecuteNonQuery`
- `ExecuteScalar`

An application should not bind a value for output parameters; it is the responsibility of ODP.NET to create the value object and populate the `OracleParameter.Value` property with the object.

When binding by position (default) to a function, ODP.NET expects the return value to be bound first, before any other parameters.

This section describes the following:

- [OracleDbType Enumeration Type](#)
- [Inference of DbType, OracleDbType, and .NET Types](#)
- [PL/SQL Associative Array Binding](#)
- [Array Binding](#)

See Also

["OracleDbType Enumeration"](#)

Command Timeouts

The `OracleCommand.CommandTimeout` property limits how long a command is allowed to execute before terminating with an exception. This setting prevents long running commands from consuming excessive resources or from blocking other necessary operations from occurring.

The database server can be interrupted via either TCP/IP urgent data or normal TCP/IP data, called out of band (OOB) or in band data, respectively. Windows-based database servers only support in band breaks, whereas all other (predominantly UNIX-based) database servers can support OOB or in band breaks.

ODP.NET, Managed Driver uses OOB breaks by default with database servers that support it. For certain network topologies, the routers or firewalls involved in the route to the database may have been configured to drop urgent data or in band the data. If the routers or firewalls can not be changed to handle urgent data appropriately, then the ODP.NET, Managed Driver can be configured to utilize in band breaks by setting the .NET configuration parameter `Disable_Oob` to on.

See Also

[settings section](#) in the [Oracle Data Provider for .NET, Managed Driver Configuration](#) section for more information.

OracleDbType Enumeration Type

`OracleDbType` enumerated values are used to explicitly specify the `OracleDbType` value of an `OracleParameter` object.

[Table 3-15](#) lists all the `OracleDbType` enumeration values with a description of each enumerated value.

Table 3-15 OracleDbType Enumeration Values

Member Name	Description
Array	Oracle Collection (VArray or Nested Table)
BFile	Oracle BFILE type
BinaryFloat	Oracle BINARY_FLOAT type
BinaryDouble	Oracle BINARY_DOUBLE type
Blob	Oracle BLOB type
Boolean	Oracle BOOLEAN type
Byte	byte type
Char	Oracle CHAR type
Clob	Oracle CLOB type
Date	Oracle DATE type
Decimal	Oracle NUMBER type
Double	8-byte FLOAT type
Int16	2-byte INTEGER type
Int32	4-byte INTEGER type
Int64	8-byte INTEGER type
IntervalDS	Oracle INTERVAL DAY TO SECOND type
IntervalYM	Oracle INTERVAL YEAR TO MONTH type
Json	Oracle JSON type
Long	Oracle LONG type
LongRaw	Oracle LONG RAW type
NChar	Oracle NCHAR type
Object	Oracle Object type

Table 3-15 (Cont.) OracleDbType Enumeration Values

Member Name	Description
NClob	Oracle NCLOB type
NVarChar2	Oracle NVARCHAR2 type
Raw	Oracle RAW type
Ref	Oracle REF type
RefCursor	Oracle REF CURSOR type
Single	4-byte FLOAT type
TimeStamp	Oracle TIMESTAMP type
TimeStampLTZ	Oracle TIMESTAMP WITH LOCAL TIME ZONE type
TimeStampTZ	Oracle TIMESTAMP WITH TIME ZONE type
Varchar2	Oracle VARCHAR2 type
XmlType	Oracle XMLType type

Note

PL/SQL LONG, LONG RAW, RAW, and VARCHAR data types can be bound with a size up to 32512 bytes.

Inference of DbType, OracleDbType, and .NET Types

This section explains the inference from the `System.Data.DbType`, `OracleDbType`, and `Value` properties in the `OracleParameter` class.

In the `OracleParameter` class, `DbType`, `OracleDbType`, and `Value` properties are linked. Specifying the value of any of these properties infers the value of one or more of the other properties.

Inference of DbType from OracleDbType

In the `OracleParameter` class, specifying the value of `OracleDbType` infers the value of `DbType` as shown in [Table 3-16](#).

Table 3-16 Inference of System.Data.DbType from OracleDbType

OracleDbType	System.Data.DbType
Array	Object
BFile	Object
Blob	Object
BinaryFloat	Single
BinaryDouble	Double
Boolean	Boolean
Byte	Byte

Table 3-16 (Cont.) Inference of System.Data.DbType from OracleDbType

OracleDbType	System.Data.DbType
Char	StringFixedLength
Clob	Object
Date	Date
Decimal	Decimal
Double	Double
Int16	Int16
Int32	Int32
Int64	Int64
IntervalDS	Object
IntervalYM	Int64
Json	String
Long	String
LongRaw	Binary
NChar	StringFixedLength
NClob	Object
NVarchar2	String
Object	Object
Raw	Binary
Ref	Object
RefCursor	Object
Single	Single
TimeStamp	DateTime
TimeStampLTZ	DateTime
TimeStampTZ	DateTimeOffset
Varchar2	String
XmlType	String

Inference of OracleDbType from DbType

In the `OracleParameter` class, specifying the value of `DbType` infers the value of `OracleDbType` as shown in [Table 3-17](#).

Table 3-17 Inference of OracleDbType from DbType

System.Data.DbType	OracleDbType
Binary	Raw
Boolean	Boolean
Byte	Byte
Currency	<i>Not Supported</i>

Table 3-17 (Cont.) Inference of OracleDbType from DbType

System.Data.DbType	OracleDbType
Date	Date
DateTime	TimeStamp
DateTimeOffset	TimeStampTZ
Decimal	Decimal
Double	Double
Guid	Blob
Int16	Int16
Int32	Int32
Int64	Int64
Object	Object
Sbyte	<i>Not Supported</i>
Single	Single
String	Varchar2
StringFixedLength	Char
Time	TimeStamp
UInt16	<i>Not Supported</i>
UInt32	<i>Not Supported</i>
UInt64	<i>Not Supported</i>
VarNumeric	<i>Not Supported</i>

Inference of DbType and OracleDbType from Value

In the `OracleParameter` class, `Value` is an object type that can be of any .NET Framework data type or ODP.NET type. If the `OracleDbType` and `DbType` properties of the `OracleParameter` class are not specified, the `OracleDbType` property is inferred from the type of the `Value` property.

[Table 3-18](#) shows the inference of `DbType` and `OracleDbType` properties from the `Value` property when the type of `Value` is one of the .NET Framework data types.

Table 3-18 Inference of DbType and OracleDbType from Value (.NET Datatypes)

Value (.NET Datatypes)	System.Data.DbType	OracleDbType
Boolean	Boolean	Boolean
Byte	Byte	Byte
Byte[]	Binary	Raw
Char / Char []	String	Varchar2
DateTime	DateTime	TimeStamp
DateTimeOffset	DateTimeOffset	TimeStampTZ
Decimal	Decimal	Decimal

Table 3-18 (Cont.) Inference of DbType and OracleDbType from Value (.NET Datatypes)

Value (.NET Datatypes)	System.Data.DbType	OracleDbType
Double	Double	Double
Float	Single	Single
Guid	Guid	Blob
Int16	Int16	Int16
Int32	Int32	Int32
Int64	Int64	Int64
IOracleCustomType	Object	Object
Single	Single	Single
String	String	Varchar2
TimeSpan	Object	IntervalDS

Note

Using other .NET Framework data types as values for the OracleParameter class without specifying either the DbType or the OracleDbType properties raises an exception because inferring DbType and OracleDbType properties from other .NET Framework data types is not supported.

[Table 3-19](#) shows the inference of DbType and OracleDbType properties from the Value property when type of Value is one of Oracle.DataAccess.Types.

Table 3-19 Inference of DbType and OracleDbType from Value (ODP.NET Types)

Value (Oracle.DataAccess.Types)	System.Data.DbType	OracleDbType
OracleBFile	Object	BFile
OracleBinary	Binary	Raw
OracleBlob	Object	Blob
OracleBoolean	Boolean	Boolean
OracleClob	Object	Clob
OracleDate	Date	Date
OracleDecimal	Decimal	Decimal
OracleIntervalDS	Object	IntervalDS
OracleIntervalYM	Int64	IntervalYM
OracleRef	Object	Ref
OracleRefCursor	Object	RefCursor
OracleString	String	Varchar2
OracleTimeStamp	DateTime	TimeStamp
OracleTimeStampLTZ	DateTime	TimeStampLTZ
OracleTimeStampTZ	DateTimeOffset	TimeStampTZ

Table 3-19 (Cont.) Inference of DbType and OracleDbType from Value (ODP.NET Types)

Value (Oracle.DataAccess.Types)	System.Data.DbType	OracleDbType
OracleXmlType	String	XmlType

PL/SQL Associative Array Binding

ODP.NET supports PL/SQL Associative Arrays (formerly known as PL/SQL Index-By Tables) binding.

An application can bind an `OracleParameter` object, as a PL/SQL Associative Array, to a PL/SQL stored procedure. The following `OracleParameter` properties are used for this feature:

- **CollectionType**
This property must be set to `OracleCollectionType.PLSQLAssociativeArray` to bind a PL/SQL Associative Array.
- **ArrayBindSize**
This property is ignored for the fixed-length element types (such as `Int32`).
For variable-length element types (such as `Varchar2`), each element in the `ArrayBindSize` property specifies the size of the corresponding element in the `Value` property.
For Output parameters, InputOutput parameters, and return values, this property must be set for variable-length variables.
If the database server supports up to 32 KB `VARCHAR2`, then each ODP.NET array element can store up to 32 KB characters or binary data. If the database server supports up to 4 KB `VARCHAR2`, then each ODP.NET array element can store up to 4 KB characters or 2 KB binary data.
- **ArrayBindStatus**
This property specifies the execution status of each element in the `OracleParameter.Value` property.
- **Size**
This property specifies the maximum number of elements to be bound in the PL/SQL Associative Array.
- **Value**
This property must be set to an array of values, null, or the `DBNull.Value` property.

ODP.NET supports binding parameters of PL/SQL Associative Arrays which contain the following data types.

- `BINARY_FLOAT`
- `CHAR`
- `DATE`
- `NCHAR`
- `NUMBER`
- `NVARCHAR2`
- `RAW`

- ROWID
- UROWID
- VARCHAR2

Using unsupported data types with associative arrays can cause an ORA-600 error.

Example of PL/SQL Associative Arrays

This example binds three OracleParameter objects as PL/SQL Associative Arrays: Param1 as an In parameter, Param2 as an InputOutput parameter, and Param3 as an Output parameter.

PL/SQL Package: MYPACK

```

/* Setup the tables and required PL/SQL:

connect scott/tiger@oracle
CREATE TABLE T1(COL1 number, COL2 varchar2(20));

CREATE or replace PACKAGE MYPACK AS
  TYPE AssocArrayVarchar2_t is table of VARCHAR(20) index by BINARY_INTEGER;
  PROCEDURE TestVarchar2(
    Param1 IN      AssocArrayVarchar2_t,
    Param2 IN OUT AssocArrayVarchar2_t,
    Param3      OUT AssocArrayVarchar2_t);
END MYPACK;
/

CREATE or REPLACE package body MYPACK as
  PROCEDURE TestVarchar2(
    Param1 IN      AssocArrayVarchar2_t,
    Param2 IN OUT AssocArrayVarchar2_t,
    Param3      OUT AssocArrayVarchar2_t)
  IS
    i integer;
  BEGIN
    -- copy a few elements from Param2 to Param1\n
    Param3(1) := Param2(1);
    Param3(2) := NULL;
    Param3(3) := Param2(3);
    -- copy all elements from Param1 to Param2\n
    Param2(1) := Param1(1);
    Param2(2) := Param1(2);
    Param2(3) := Param1(3);
    -- insert some values to db\n
    FOR i IN 1..3 LOOP
      insert into T1 values(i,Param2(i));
    END LOOP;
  END TestVarchar2;
END MYPACK;
/
*/

// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;

class AssociativeArraySample
{
  static void Main()

```

```
{
    OracleConnection con = new OracleConnection();

    con.ConnectionString = "User Id=scott;Password=tiger;Data Source=oracle";
    con.Open();
    Console.WriteLine("Connected to Oracle" + con.ServerVersion);

    OracleCommand cmd = new OracleCommand(
        "begin MyPack.TestVarchar2(:1, :2, :3); end;", con);

    OracleParameter Param1 = cmd.Parameters.Add("1", OracleDbType.Varchar2);
    OracleParameter Param2 = cmd.Parameters.Add("2", OracleDbType.Varchar2);
    OracleParameter Param3 = cmd.Parameters.Add("3", OracleDbType.Varchar2);

    Param1.Direction = ParameterDirection.Input;
    Param2.Direction = ParameterDirection.InputOutput;
    Param3.Direction = ParameterDirection.Output;

    // Specify that we are binding PL/SQL Associative Array
    Param1.CollectionType = OracleCollectionType.PLSQLAssociativeArray;
    Param2.CollectionType = OracleCollectionType.PLSQLAssociativeArray;
    Param3.CollectionType = OracleCollectionType.PLSQLAssociativeArray;

    // Setup the values for PL/SQL Associative Array
    Param1.Value = new string[3] {
        "First Element", "Second Element ", "Third Element "
    };
    Param2.Value = new string[3] {
        "First Element", "Second Element ", "Third Element "
    };
    Param3.Value = null;

    // Specify the maximum number of elements in the PL/SQL Associative Array
    Param1.Size = 3;
    Param2.Size = 3;
    Param3.Size = 3;

    // Setup the ArrayBindSize for Param1
    Param1.ArrayBindSize = new int[3] { 13, 14, 13 };

    // Setup the ArrayBindStatus for Param1
    Param1.ArrayBindStatus = new OracleParameterStatus[3] {
        OracleParameterStatus.Success, OracleParameterStatus.Success,
        OracleParameterStatus.Success};

    // Setup the ArrayBindSize for Param2
    Param2.ArrayBindSize = new int[3] { 20, 20, 20 };

    // Setup the ArrayBindSize for Param3
    Param3.ArrayBindSize = new int[3] { 20, 20, 20 };

    // execute the cmd
    cmd.ExecuteNonQuery();

    //print out the parameter's values
    Console.WriteLine("parameter values after executing the PL/SQL block");
    for (int i = 0; i < 3; i++)
        Console.WriteLine("Param2[{0}] = {1} ", i,
            (cmd.Parameters[1].Value as Array).GetValue(i));

    for (int i = 0; i < 3; i++)
        Console.WriteLine("Param3[{0}] = {1} ", i,
```

```

        (cmd.Parameters[2].Value as Array).GetValue(i));

    // Close and Dispose OracleConnection object
    con.Close();
    con.Dispose();
    Console.WriteLine("Disconnected");
}
}

```

Array Binding

The array bind feature enables applications to bind arrays of a type using the `OracleParameter` class. Using the array bind feature, an application can insert multiple rows into a table in a single database round-trip.

The following example inserts three rows into the `Dept` table with a single database round-trip. The `OracleCommand` `ArrayBindCount` property defines the number of elements of the array to use when executing the statement.

```

// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;

class ArrayBindSample
{
    static void Main()
    {
        OracleConnection con = new OracleConnection();
        con.ConnectionString = "User Id=scott;Password=tiger;Data Source=oracle;";
        con.Open();
        Console.WriteLine("Connected successfully");

        int[] myArrayDeptNo = new int[3] { 10, 20, 30 };
        OracleCommand cmd = new OracleCommand();

        // Set the command text on an OracleCommand object
        cmd.CommandText = "insert into dept(deptno) values (:deptno)";
        cmd.Connection = con;

        // Set the ArrayBindCount to indicate the number of values
        cmd.ArrayBindCount = 3;

        // Create a parameter for the array operations
        OracleParameter prm = new OracleParameter("deptno", OracleDbType.Int32);

        prm.Direction = ParameterDirection.Input;
        prm.Value = myArrayDeptNo;

        // Add the parameter to the parameter collection
        cmd.Parameters.Add(prm);

        // Execute the command
        cmd.ExecuteNonQuery();
        Console.WriteLine("Insert Completed Successfully");

        // Close and Dispose OracleConnection object
        con.Close();
        con.Dispose();
    }
}

```

```
}  
}
```

See Also

["Value"](#) for more information

OracleParameter Array Bind Properties

The `OracleParameter` class provides two properties for granular control when using the array bind feature:

- `ArrayBindSize`

The `ArrayBindSize` property is an array of integers specifying the maximum size for each corresponding value in an array. The `ArrayBindSize` property is similar to the `Size` property of an `OracleParameter` object, except the `ArrayBindSize` property specifies the size for each value in an array.

Before the execution, the application must populate the `ArrayBindSize` property; after the execution, ODP.NET populates it.

The `ArrayBindSize` property is used only for parameter types that have variable length such as `Clob`, `Blob`, and `Varchar2`. The size is represented in bytes for binary data types, and characters for the Unicode string types. The count for string types does not include the terminating character. The size is inferred from the actual size of the value, if it is not explicitly set. For an output parameter, the size of each value is set by ODP.NET. The `ArrayBindSize` property is ignored for fixed-length data types.

The maximum `ArrayBindSize` size is 2 GB for both character and binary data.

- `ArrayBindStatus`

The `ArrayBindStatus` property is an array of `OracleParameterStatus` values that specify the status of each corresponding value in an array for a parameter. This property is similar to the `Status` property of the `OracleParameter` object, except that the `ArrayBindStatus` property specifies the status for each array value.

Before the execution, the application must populate the `ArrayBindStatus` property. After the execution, ODP.NET populates the property. Before the execution, an application using the `ArrayBindStatus` property can specify a `NULL` value for the corresponding element in the array for a parameter. After the execution, ODP.NET populates the `ArrayBindStatus` property, indicating whether the corresponding element in the array has a `null` value, or if data truncation occurred when the value was fetched.

Error Handling for Array Binding

If an error occurs during an array bind execution, it can be difficult to determine which element in the `Value` property caused the error. ODP.NET provides a way to determine the row where the error occurred, making it easier to find the element in the row that caused the error.

When an `OracleException` object is thrown during an array bind execution, the `OracleErrorCollection` object contains one or more `OracleError` objects. Each of these `OracleError` objects represents an individual error that occurred during the execution, and contains a provider-specific property, `ArrayBindIndex`, which indicates the row number at which the error occurred.

The following example demonstrates error handling for array binding:

```
/* Database Setup
connect scott/tiger@oracle
drop table depttest;
create table depttest(deptno number(2));
*/

// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;

class ArrayBindExceptionSample
{
    static void Main()
    {
        OracleConnection con = new OracleConnection();
        con.ConnectionString = "User Id=scott;Password=tiger;Data Source=oracle;";
        con.Open();

        OracleCommand cmd = new OracleCommand();

        // Start a transaction
        OracleTransaction txn = con.BeginTransaction(IsolationLevel.ReadCommitted);

        try
        {
            int[] myArrayDeptNo = new int[3] { 10, 200000, 30 };
            // int[] myArrayDeptNo = new int[3]{ 10,20,30};

            // Set the command text on an OracleCommand object
            cmd.CommandText = "insert into depttest(deptno) values (:deptno)";
            cmd.Connection = con;

            // Set the ArrayBindCount to indicate the number of values
            cmd.ArrayBindCount = 3;

            // Create a parameter for the array operations
            OracleParameter prm = new OracleParameter("deptno", OracleDbType.Int32);

            prm.Direction = ParameterDirection.Input;
            prm.Value = myArrayDeptNo;

            // Add the parameter to the parameter collection
            cmd.Parameters.Add(prm);

            // Execute the command
            cmd.ExecuteNonQuery();
        }
        catch (OracleException e)
        {
            Console.WriteLine("OracleException {0} occurred", e.Message);
            if (e.Number == 24381)
            {
                for (int i = 0; i < e.Errors.Count; i++)
                {
                    Console.WriteLine("Array Bind Error {0} occurred at Row Number {1}",
                        e.Errors[i].Message, e.Errors[i].ArrayBindIndex);
                }
            }

            txn.Commit();
        }
        cmd.Parameters.Clear();
        cmd.CommandText = "select count(*) from depttest";
    }
}
```

```
decimal rows = (decimal)cmd.ExecuteScalar();

Console.WriteLine("{0} row have been inserted", rows);
con.Close();
con.Dispose();
}
}
```

See Also

["ArrayBindIndex"](#) for more information

OracleParameterStatus Enumeration Types

[Table 3-20](#) lists OracleParameterStatus enumeration values.

Table 3-20 OracleParameterStatus Members

Member Names	Description
Success	For input parameters, indicates that the input value has been assigned to the column. For output parameters, indicates that the provider assigned an intact value to the parameter.
NullFetched	Indicates that a NULL value has been fetched from a column or an OUT parameter.
NullInsert	Indicates that a NULL value is to be inserted into a column.
Truncation	Indicates that truncation has occurred when fetching the data from the column.

Batch Processing

The `OracleDataAdapter.UpdateBatchSize` property enables batch processing when the `OracleDataAdapter.Update` method is called. `UpdateBatchSize` is a numeric property that indicates how many `DataSet` rows to update the Oracle database for each round-trip.

This enables the developer to reduce the number of round-trips to the database.

See Also

["UpdateBatchSize"](#)

Statement Caching

Statement caching eliminates the need to parse each SQL or PL/SQL statement before execution by caching server cursors created during the initial statement execution. Subsequent executions of the same statement can reuse the parsed information from the cursor, and then execute the statement without reparsing, for better performance.

In order to see performance gains from statement caching, Oracle recommends caching only those statements that will be repeatedly executed. Furthermore, SQL or PL/SQL statements

should use parameters rather than literal values. Doing so takes full advantage of statement caching, because parsed information from parameterized statements can be reused even if the parameter values change in subsequent executions. However, if the literal values in the statements are different, the parsed information cannot be reused unless the subsequent statements also have the same literal values.

Statement Caching Connection String Attributes

The following connection string attributes control the behavior of the ODP.NET statement caching feature:

- **Statement Cache Size**

This attribute enables or disables ODP.NET statement caching. By default, this attribute is set to 0 (disabled). If it is set to a value greater than 0, ODP.NET statement caching is enabled and the value specifies the maximum number of statements that can be cached for a connection. Once a connection has cached up to the specified maximum cache size, the least recently used cursor is freed to make room to cache the newly created cursor.

If self tuning is enabled, then statement caching is enabled as well. The Statement Cache Size is configured automatically in such cases.

- **Statement Cache Purge**

This attribute provides a way for connections to purge all statements that are cached when a connection is closed or placed back into the connection pool. By default, this attribute is set to false, which means that cursors are not freed when connections are placed back into the pool.

Enabling Statement Caching through the Registry

To enable statement caching by default for all ODP.NET applications running in a system, without changing the application, set the registry key of `HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\ODP.NET\Assembly_Version\StatementCacheSize` to a value greater than 0. This value specifies the number of cursors that are to be cached on the server.

The default value for the system can be overridden at the connection pool level. The Statement Cache Size attribute can be set to a different size than the registry value or it can be turned off. The Statement Cache Size can also be configured through an XML configuration file.

See Also

[Configuring Oracle Data Provider for .NET](#) for more details.

Statement Caching Methods and Properties

The following property and method are relevant only when statement caching is enabled:

- `OracleCommand.AddToStatementCache` property

If statement caching is enabled, having this property set to true (default) adds statements to the cache when they are executed. If statement caching is disabled or if this property is set to false, the executed statement is not cached.

- `OracleConnection.PurgeStatementCache` method

This method purges all the cached statements by closing all open cursors on the database that are associated with the particular connection. Note that statement caching remains enabled after this call.

Connections and Statement Caching

Statement caching is managed separately for each connection. Therefore, executing the same statement on different connections requires parsing once for each connection and caching a separate cursor for each connection.

Pooling and Statement Caching

Pooling and statement caching can be used in conjunction. If connection pooling is enabled and the Statement Cache Purge attribute is set to false, statements executed on each separate connection are cached throughout the lifetime of the pooled connection.

If the Statement Cache Purge attribute is set to true, all the cached cursors are freed when the connection is placed back into the pool. When connection pooling is disabled, cursors are cached during the lifetime of the connection, but the cursors are closed when the `OracleConnection` object is closed or disposed of.

If schema object metadata is changed, such as dropping a table column, while the .NET app is running, ODP.NET should not use the cached schema information. The cache is now out of date and returns an "ORA-01007 variable not in select list" or similar error. If your app changes schema objects during the .NET app's running lifetime, then disable both statement caching and metadata pooling for ODP.NET commands that use those schema objects to avoid the error.

Self-Tuning

ODP.NET applications can be self-tuned for performance optimization. ODP.NET dynamically monitors application queries during runtime.

Note

Self-tuning for applications does not take place if the `Pooling` connection string attribute is set to false.

The statement cache size ([StatementCacheSize](#)) is tuned automatically by monitoring the statements that are executed by the application. The following sections discuss self-tuning in applications:

- [Self-Tuning Statement Caching](#)
- [Enabling or Disabling Self-Tuning for Applications](#)
- [Tracing Optimization Changes](#)

Self-Tuning Statement Caching

Statement caching helps improve performance by eliminating the need to re-parse each SQL or PL/SQL statement before execution.

If self-tuning is enabled for an application, then ODP.NET continuously monitors application behavior in order to determine the optimum value for the statement cache size. Any statement cache size value specified in the connection string, configuration file, or registry is ignored.

When the application first initializes, it uses the default value of statement cache size. As the application executes statements, ODP.NET collects statistics that are used to self-tune the value of statement cache size. Self-tuning of statement cache size results in increased performance.

Note

To take full advantage of statement caching, you should not dynamically generate statements, with different inline values, for every statement execution. Instead, use parameterized commands to minimize the number of unique statements that need to be executed and cached. This is because only one statement needs to be cached for every unique command text, regardless of the parameter values and the number of times that the statement is executed.

The maximum number of statements that can be cached per connection is determined by the `MaxStatementCacheSize` configuration attribute. The `MaxStatementCacheSize` value can be specified in the Windows registry or XML configuration file.

The `MaxStatementCacheSize` setting is useful in limiting the number of cached statements, as well as the number of open cursors. This is because a cached statement equates to a cursor being opened on the server. For this reason, you should not set `MaxStatementCacheSize` to a value that is greater than the database `OPEN_CURSORS` setting.

The following Windows registry key is used to configure the `MaxStatementCacheSize` configuration attribute:

```
HKLM\Software\Oracle\ODP.NET\version\MaxStatementCacheSize
```

The `MaxStatementCacheSize` key is of type `REG_SZ`. It can be set to an integer value between 0 and `System.Int32.MaxValue`.

The following example sets the `MaxStatementCacheSize` property in an ADO.NET 2.0, or above, configuration file:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <oracle.dataaccess.client>
    <settings>
      <add name="MaxStatementCacheSize" value="300"/>
    </settings>
  </oracle.dataaccess.client>
</configuration>
```

If self-tuning is disabled for an application, then the value of statement cache size is determined by the settings in the connection string, configuration file, or the registry. If statement cache size is not specified in any of these sources, then the default value of statement cache size is set to 0. To have ODP.NET configured with the same default settings as previous releases of ODP.NET, disable self-tuning and set the `StatementCacheSize` value to 10.

① See Also

["Statement Caching"](#)

Enabling or Disabling Self-Tuning for Applications

Self-tuning for ODP.NET applications is enabled by default. An application can enable or disable self-tuning using one of the following methods:

- Self-Tuning Connection String Attribute

An application can modify the Self Tuning connection string attribute to enable or disable self-tuning for a particular connection pool. The default value for Self Tuning is true.

- Windows Registry

An application can enable or disable self-tuning for a particular version of ODP.NET by modifying the following registry entry:

HKLM\Software\Oracle\ODP.NET\version\SelfTuning

The SelfTuning key is of type REG_SZ. It can be set to either 1 (enabled) or 0 (disabled).

- Configuration File

An ODP.NET application can modify the application configuration file (app.config) or Web configuration file (web.config) to enable or disable self-tuning.

The following example shows how to enable self-tuning in an ADO.NET 2.0 application configuration file:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <oracle.dataaccess.client>
    <settings>
      <add name="SelfTuning" value="1"/>
    </settings>
  </oracle.dataaccess.client>
</configuration>
```

① Note

If the optimal statement cache size is known for an application, then you can disable self-tuning and set StatementCacheSize to its optimum value in the registry, configuration file, or the application. If self-tuning is disabled and StatementCacheSize is not set at all, then the default value of 0 is used for StatementCacheSize.

Tracing Optimization Changes

Applications can trace optimization changes made by self-tuning. All changes to StatementCacheSize are traced. Errors, if any, are also traced.

The TraceLevel used for tracing self-tuning is 64 for unmanaged ODP.NET. Managed ODP.NET and ODP.NET Core uses TraceLevel to 7.

① See Also

[Table 2-3](#) for details on `TraceLevel` values

Data Transmission Performance

ODP.NET data transmission performance can be optimized via the Session Data Unit (SDU) size setting. The SDU is a buffer that Oracle uses to place data before transmitting it across the network. Oracle sends the data in the buffer either when requested or when it is full.

Adjusting the size of the SDU buffers relative to the amount of data to send at any one time can improve performance, network utilization, and memory consumption. When large amounts of data are being transmitted, increasing the SDU size can improve performance and network throughput. SDU size can be adjusted lower or higher to achieve higher throughput for a specific deployment. A larger SDU size allows more data to be sent per buffer, reducing number of database round trips for large amounts of data. To accommodate the larger buffer, more memory consumption occurs.

This buffer size can be set on either the ODP.NET client side or database server side. ODP.NET supports SDU sizes between 256 and 2,097,152 bytes. In ODP.NET, this can be set in the .NET configuration file (.NET Framework), `tnsnames.ora`, or `sqlnet.ora`.

For managed ODP.NET and ODP.NET Core, if the application does not provide a client side SDU value, then connections will use the database server's SDU size. If the client side SDU size is set, then connections will compare it against the DB server's SDU size, and then choose the lower of the two values for the SDU size to use.

Unmanaged ODP.NET SDU setting behavior is the same as the Oracle database client's.

① See Also

Optimizing Performance in *Oracle AI Database Net Services Administrator's Guide* for more SDU details.

Securing Identifiers and Literals from SQL Injection Attacks

ODP.NET command parameter binding is recommended to protect against many SQL injection attack types. In cases in which literal or database identifier values cannot be bound as parameters, developers historically had to then validate those input values programmatically. In essence, their code took on the onus of SQL injection protection.

Managed ODP.NET and ODP.NET Core 23.8 introduce methods to make securing ODP.NET app code easier when using these literal and identifier values. These methods sanitize the input values to prevent their usage as SQL injection vectors.

The new methods validate literals and database identifiers, and then add quotes around them if they are missing.

Database identifiers include Oracle SQL reserved words.

Note

- [OracleDBMSAssert Class](#)
- Oracle SQL Reserved Words

ODP.NET Types Overview

ODP.NET types represent Oracle native data types and PL/SQL data types as a structure or as a class. ODP.NET type structures follow [value semantics](#), while ODP.NET type classes follow [reference semantics](#). ODP.NET types provide safer and more efficient ways of obtaining Oracle native data and PL/SQL data types in a .NET application than .NET types. For example, an `OracleDecimal` structure holds up to 38 digits of precision, while a .NET `Decimal` only holds up to 28.

[Table 3-21](#) lists data types supported by ODP.NET and their corresponding ODP.NET types: data types in the first column refer to both Oracle native data types and PL/SQL data types of that name. Those data types that exist only in PL/SQL are indicated by (PL/SQL only) after the data type name. The entries for the PL/SQL data types also represent the subtypes of the data types, if any. The third column lists the .NET Framework data type that corresponds to the `Value` property of each ODP.NET type.

Table 3-21 Value Property Type of ODP.NET Type

Oracle Native Data Type or PL/SQL Data Type	ODP.NET Type	.NET Framework Data Types
BFILE	<code>OracleBFile</code> class	<code>System.Byte[]</code>
BINARY_DOUBLE	<code>OracleDecimal</code> structure	<code>System.Decimal</code>
BINARY_FLOAT	<code>OracleDecimal</code> structure	<code>System.Decimal</code>
BINARY_INTEGER (PL/SQL only)	<code>OracleDecimal</code> structure	<code>System.Decimal</code>
BLOB	<code>OracleBlob</code> class	<code>System.Byte[]</code>
BOOLEAN	<code>OracleBoolean</code> structure	<code>System.Boolean</code>
CHAR	<code>OracleString</code> structure	<code>System.String</code>
CLOB	<code>OracleClob</code> class	<code>System.String</code>
DATE	<code>OracleDate</code> structure	<code>System.DateTime</code>
INTERVAL DAY TO SECOND	<code>OracleIntervalDS</code> structure	<code>System.TimeSpan</code>
INTERVAL YEAR TO MONTH	<code>OracleIntervalYM</code> structure	<code>System.Int64</code>
JSON	<code>OracleString</code> structure	<code>System.String</code>
LONG	<code>OracleString</code> structure	<code>System.String</code>
LONG RAW	<code>OracleBinary</code> structure	<code>System.Byte[]</code>
NCHAR	<code>OracleString</code> structure	<code>System.String</code>
NCLOB	<code>OracleClob</code> class	<code>System.String</code>
NUMBER	<code>OracleDecimal</code> structure	<code>System.Decimal</code>
NVARCHAR2	<code>OracleString</code> structure	<code>System.String</code>

Table 3-21 (Cont.) Value Property Type of ODP.NET Type

Oracle Native Data Type or PL/SQL Data Type	ODP.NET Type	.NET Framework Data Types
PLS_INTEGER (PL/SQL only)	OracleDecimal structure	System.Decimal
RAW	OracleBinary structure	System.Byte[]
REF	OracleRef class	System.String
REF CURSOR	OracleRefCursor class	System.String
ROWID	OracleString structure	System.String
TIMESTAMP	OracleTimeStamp structure	System.DateTime
TIMESTAMP WITH LOCAL TIME ZONE	OracleTimeStampLTZ structure	System.DateTime
TIMESTAMP WITH TIME ZONE	OracleTimeStampTZ structure	System.DateTimeOffset
UROWID	OracleString structure	System.String
VARCHAR2	OracleString structure	System.String
VECTOR	OracleString structure	System.Array
XMLType	OracleXmlType class	System.String

Deserializing ODP.NET Types into DataSet and DataTable

Due to a change in all .NET versions to enhance application security, the allowed DataSet and DataTable data types that can be deserialized are now restricted. This change applies to .NET 5, plus .NET Core and .NET Framework updates. If your DataSets and DataTables use Oracle data types with one of these new .NET versions, then you will have to add ODP.NET-specific data types to the “allow” list so that they can be deserialized into DataSet or DataTable. If an attempt is made to deserialize ODP.NET-specific types without adding them to the allow list, an ODP.NET type initializer exception will be encountered.

The most straightforward way to add all ODP.NET data types to the allow list to call the OracleConfiguration AddOracleTypesDeserialization method in your application.

Alternatively, in .NET Framework 4, the specific ODP.NET data types can be added individually to the application .NET configuration file. Here's a sample configuration file for adding all managed ODP.NET 21c data types to the allow list.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <sectionGroup name="system.data.dataset.serialization"
type="System.Data.SerializationSettingsSectionGroup, System.Data, Version=4.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089">
      <section name="allowedTypes" type="System.Data.AllowedTypesSectionHandler,
System.Data, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
    </sectionGroup>
  </configSections>
  <system.data.dataset.serialization>
    <allowedTypes>
      <!-- <add type="assembly qualified type name" /> -->
      <add type="Oracle.ManagedDataAccess.Types.OracleBinary,
Oracle.ManagedDataAccess, Version=4.122.21.1, Culture=neutral,
PublicKeyToken=89b483f429c47342" />
    </allowedTypes>
  </system.data.dataset.serialization>
</configuration>
```

```

    <add type="Oracle.ManagedDataAccess.Types.OracleBlob, Oracle.ManagedDataAccess,
Version=4.122.21.1, Culture=neutral, PublicKeyToken=89b483f429c47342" />
    <add type="Oracle.ManagedDataAccess.Types.OracleClob, Oracle.ManagedDataAccess,
Version=4.122.21.1, Culture=neutral, PublicKeyToken=89b483f429c47342" />
    <add type="Oracle.ManagedDataAccess.Types.OracleDate, Oracle.ManagedDataAccess,
Version=4.122.21.1, Culture=neutral, PublicKeyToken=89b483f429c47342" />
    <add type="Oracle.ManagedDataAccess.Types.OracleDecimal, Oracle.ManagedDataAccess,
Version=4.122.21.1, Culture=neutral, PublicKeyToken=89b483f429c47342" />
    <add type="Oracle.ManagedDataAccess.Types.OracleIntervalDS,
Oracle.ManagedDataAccess, Version=4.122.21.1, Culture=neutral,
PublicKeyToken=89b483f429c47342" />
    <add type="Oracle.ManagedDataAccess.Types.OracleIntervalYM,
Oracle.ManagedDataAccess, Version=4.122.21.1, Culture=neutral,
PublicKeyToken=89b483f429c47342" />
    <add type="Oracle.ManagedDataAccess.Types.OracleRef, Oracle.ManagedDataAccess,
Version=4.122.21.1, Culture=neutral, PublicKeyToken=89b483f429c47342" />
    <add type="Oracle.ManagedDataAccess.Types.OracleString, Oracle.ManagedDataAccess,
Version=4.122.21.1, Culture=neutral, PublicKeyToken=89b483f429c47342" />
    <add type="Oracle.ManagedDataAccess.Types.OracleTimeStamp,
Oracle.ManagedDataAccess, Version=4.122.21.1, Culture=neutral,
PublicKeyToken=89b483f429c47342" />
    <add type="Oracle.ManagedDataAccess.Types.OracleTimeStampLTZ,
Oracle.ManagedDataAccess, Version=4.122.21.1, Culture=neutral,
PublicKeyToken=89b483f429c47342" />
    <add type="Oracle.ManagedDataAccess.Types.OracleTimeStampTZ,
Oracle.ManagedDataAccess, Version=4.122.21.1, Culture=neutral,
PublicKeyToken=89b483f429c47342" />
    <add type="Oracle.ManagedDataAccess.Types.OracleXmlType, Oracle.ManagedDataAccess,
Version=4.122.21.1, Culture=neutral, PublicKeyToken=89b483f429c47342" />
    <!-- additional <add /> elements as needed -->
  </allowedTypes>
</system.data.dataset.serialization>
</configuration>

```

With the .NET configuration file, developers can enable specific ODP.NET data types to allow, rather than enable all of them with `AddOracleTypesDeserialization` method.

① See Also

<https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/dataset-datatable-dataview/security-guidance>

GUIDs

ODP.NET does support GUIDs. GUIDs can be inserted into a RAW(16) column which is big enough to hold any GUID value. But caution needs to be taken in order to handle GUIDs appropriately. This is due to the fact that as the .NET Guid structure flips the byte values in reverse order for the integer-based parts of the GUID values when `Guid(byte[])` constructor is used and when the `ToByteArray()` method on the Guid struct is invoked.

When the Guid is provided to ODP.NET as an input parameter, for example, ODP.NET invokes the `ToByteArray()` method before passing the values to the Oracle Database as raw bytes. This means that if the Guid originally has a sequence of bytes of

```
9D4E51F764A940E4854D667F0DD61093
```

then the byte[] representation of the GUID would be changed to

```
F7514E9DA964E440854D667F0DD61093
```

after the `ToByteArray()` method invocation that is invoked by ODP.NET internally before it's inserted into the `RAW(16)` column.

And when the same column value is then retrieved from the Oracle database, the value that is obtained is

```
F7514E9DA964E440854D667F0DD61093
```

which is how it was stored in the database. When the `GetGuid()` method is invoked on the `OracleDataReader` object, ODP.NET constructs the `Guid` structure using the `Guid(byte[ ])` constructor, which flips it back to the original sequence of bytes:

```
9D4E51F764A940E4854D667F0DD61093
```

This means that the sequence of bytes of the GUID in the application is not in the exact same order as how it is stored in the database. In other words, if the application executes a SQL using literal byte values, such as

```
select * from ... where <raw/guid_column> = '9D4E51F764A940E4854D667F0DD61093';
```

there will be no rows returned. The same GUID was inserted into the table as:

```
F7514E9DA964E440854D667F0DD61093
```

The application will be able to query for the row with the matching GUID if the application

- binds the guid value using `Guid` structure as an input parameter
- constructs the byte literal value that is to be used in the SQL from the `byte[ ]` returned from the `ToByteArray()` method invocation on the `Guid` structure.

The application developer should be cautious and mindful as to when the `Guid(byte[ ])` constructor and the `ToByteArray()` method of the `Guid` structure is called, which can alter the sequence of the bytes. A simple program below demonstrates how the integer-based parts of the GUID values are flipped when `Guid(byte[ ])` constructor and when the `ToByteArray()` method are invoked.

```
using System;
using System.Text;
using System.Data;
using Oracle.ManagedDataAccess.Client;

class T
{
    static string ByteToString(byte[] data)
    {
        StringBuilder sb = new StringBuilder(16);
        foreach (var b in data)
            sb.Append($"{b:X2}");
        return sb.ToString();
    }

    static void Main()
    {
        try
        {
            OracleConnection con = new OracleConnection("user id=<user
id>;password=<password>;data source=<data source>");
```

```
con.Open();

//
// Generate a new GUID
//
Guid guid = Guid.NewGuid();
string original = guid.ToString().ToUpper().Replace("-", "");
Console.WriteLine("Original Guid           : " + original);

//
// Drop the table
//
OracleCommand cmd = new OracleCommand("drop table test_guid_table", con);
try { cmd.ExecuteNonQuery(); } catch {}

//
// Create the table
//
cmd.CommandText = "create table test_guid_table (col1 RAW(16), col2 VARCHAR2(64))";
cmd.ExecuteNonQuery();

//
// Insert the newly generated GUID to the DB
//
cmd.CommandText = "insert into test_guid_table values (:1, 'new guid')";
cmd.Parameters.Add(string.Empty, OracleDbType.Raw);
cmd.Parameters[0].Value = guid;
cmd.ExecuteNonQuery();

//
// Query from the test table
//
cmd.CommandText = "select * from test_guid_table";
OracleDataReader reader = cmd.ExecuteReader();

while (reader.Read())
{
    //
    // Get the RAW data as byte[]
    //
    byte[] guid_byte_array = (byte[])reader.GetValue(0);
    Console.WriteLine("GetValue() as byte[] / as-is in DB: " +
ByteToString(guid_byte_array));
    Console.WriteLine();

    //
    // Get the RAW data as Guid then convert to byte[]
    //
    Guid retrieved_guid = (Guid)reader.GetGuid(0);
    byte[] retrieved_guid_byte_array = retrieved_guid.ToByteArray();
    Console.WriteLine("GetGuid() then Guid.ToString()      : " +
retrieved_guid_byte_array.ToString());
    Console.WriteLine("GetGuid() then Guid.ToByteArray() : " +
ByteToString(retrieved_guid.ToByteArray()));
}

//
// Find a matching row by binding the original GUID as-is
//
cmd.Parameters.Clear();
Console.WriteLine("\nGuid Input Parameter           : " +
ByteToString(guid.ToByteArray()));
```



```

        cmd.CommandText = "select count(*) from test_guid_table where col1 = :1";
        cmd.Parameters.Add(string.Empty, OracleDbType.Raw);
        cmd.Parameters[0].Value = guid;
        reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            Console.WriteLine("Rows found by binding GUID          : " +
(decimal)reader.GetValue(0));
        }

        //
        // Find a matching row by binding a byte[] from the original GUID
        //
        byte[] byte_array_param = guid.ToByteArray();
        Console.WriteLine("\nbyte[] Input Parameter          : " +
ByteToString(byte_array_param));
        cmd.CommandText = "select count(*) from test_guid_table where col1 = :1";
        cmd.Parameters.Clear();
        cmd.Parameters.Add(string.Empty, OracleDbType.Raw);
        cmd.Parameters[0].Value = byte_array_param;
        reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            Console.WriteLine("Rows found by binding byte[]      : " +
(decimal)reader.GetValue(0));
        }

        //
        // Find a matching row by matching the binary/raw data (inline) using
        Guid.ToByteArray()
        //
        cmd.CommandText = "select count(*) from test_guid_table where col1 = '" +
ByteToString(guid.ToByteArray()) + "'";
        Console.WriteLine("\nLiteral RAW (from byte array)      : " +
ByteToString(guid.ToByteArray()));
        cmd.Parameters.Clear();
        reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            Console.WriteLine("Rows found by inlined data        : " +
(decimal)reader.GetValue(0));
        }

        //
        // Find a matching row by matching the binary/raw data (inline) using
        Guid.ToString()
        //
        cmd.CommandText = "select count(*) from test_guid_table where col1 = '" + original
+ "'";
        Console.WriteLine("\nLiteral RAW (from string)          : " + original);
        cmd.Parameters.Clear();
        reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            Console.WriteLine("Rows found by inlined data        : " +
(decimal)reader.GetValue(0));
        }
    } catch (Exception ex) { Console.WriteLine(ex); }
}
}

```

A sample output from the sample code could be:

```

Original Guid                : D54909F4169541CFA919F6752414909F
GetValue() as byte[] / as-is in DB: F40949D59516CF41A919F6752414909F

GetGuid() then Guid.ToString() : System.Byte[]
GetGuid() then Guid.ToByteArray() : F40949D59516CF41A919F6752414909F

Guid Input Parameter        : F40949D59516CF41A919F6752414909F
Rows found by binding GUID  : 1

byte[] Input Parameter      : F40949D59516CF41A919F6752414909F
Rows found by binding byte[] : 1

Literal RAW (from byte array) : F40949D59516CF41A919F6752414909F
Rows found by inlined data    : 1

Literal RAW (from string)    : D54909F4169541CFA919F6752414909F
Rows found by inlined data    : 0

```

Note

Every execution of the test generates a new or different GUID.

Obtaining Data from an OracleDataReader Object

The `ExecuteReader` method of the `OracleCommand` object returns an `OracleDataReader` object, which is a read-only, forward-only result set.

This section provides the following information about the `OracleDataReader` object:

- [Typed OracleDataReader Accessors](#)
- [Obtaining LONG and LONG RAW Data](#)
- [Obtaining LOB Data](#)
- [Controlling the Number of Rows Fetched in One Database Round-Trip](#)

Typed OracleDataReader Accessors

The `OracleDataReader` class provides two types of typed accessors:

- [.NET Type Accessors](#)
- [ODP.NET Type Accessors](#)

.NET Type Accessors

[Table 3-22](#) lists all the Oracle native database types that ODP.NET supports, and the corresponding .NET types that can represent the Oracle native type. If more than one .NET type can be used to represent an Oracle native type, the first entry is the .NET type that best represents the Oracle native type. The third column indicates the valid typed accessor that can be invoked for an Oracle native type to be obtained as a .NET type. If an invalid typed accessor is used for a column, an `InvalidCastException` is thrown. Oracle native data types depend on the version of the database; therefore, some data types are not available in earlier versions of Oracle Database.

① See Also

- "[OracleDataAdapter Class](#) "
- "[OracleDataReader Class](#)"

Table 3-22 .NET Type Accessors

Oracle Native Data Type	.NET Type	Typed Accessor
BFILE	System.Byte[]	GetBytes
	System.Guid	GetGuid ¹
BINARY_DOUBLE	System.Double	GetDouble
	System.Boolean	GetBoolean
	System.String	GetString
BINARY_FLOAT	System.Single	GetFloat
	System.Boolean	GetBoolean
	System.String	GetString
BLOB	System.Byte[]	GetBytes
	System.Guid	GetGuid ¹
BOOLEAN	System.Boolean	GetBoolean
CHAR	System.String	GetString
	System.Char[]	GetChars
	System.Boolean	GetBoolean
CLOB	System.String	GetString
	System.Char[]	GetChars
DATE	System.DateTime	GetDateTime
	System.String	GetString ¹
FLOAT	System.Boolean	GetBoolean
	System.Byte	GetByte
	System.Decimal	GetDecimal
	System.Double	GetDouble
	System.Single	GetFloat
	System.Int16	GetInt16
	System.Int32	GetInt32
	System.Int64	GetInt64
	System.String	GetString
	System.Object	GetValue
INTERVAL DAY TO SECOND	System.Timespan	GetTimeSpan
	System.String	GetString ¹
	System.Decimal	GetDecimal ²
INTERVAL YEAR TO MONTH	System.Int64	GetInt64
	System.String	GetString ¹
	System.Decimal	GetDecimal ²

Table 3-22 (Cont.) .NET Type Accessors

Oracle Native Data Type	.NET Type	Typed Accessor
JSON	System.String	GetString
	OracleString	GetOracleString
LONG	System.String	GetString
	System.Char[]	GetChars
LONG RAW	System.Byte[]	GetBytes
	System.Guid	GetGuid ¹
NCHAR	System.String	GetString
	System.Char[]	GetChars
	System.Boolean	GetBoolean
NCLOB	System.String	GetString
	System.Char[]	GetChars
NUMBER	System.Decimal	GetDecimal
	System.Byte	GetByte
	System.Int16	GetInt16
	System.Int32	GetInt32
	System.Int64	GetInt64
	System.Single	GetFloat
	System.Double	GetDouble
	System.Boolean	GetBoolean
	System.String	GetString
	System.Char[]	GetChars
NVARCHAR2	System.String	GetString
	System.Char[]	GetChars
	System.Boolean	GetBoolean
RAW	System.Byte[]	GetBytes
	System.Guid	GetGuid ¹
REF	System.String	GetString
REF CURSOR	System.String	GetString ¹
ROWID	System.String	GetString
	System.Char[]	GetChars
TIMESTAMP	System.DateTime	GetDateTime
	System.String	GetString ¹
TIMESTAMP WITH LOCAL TIME ZONE	System.DateTime	GetDateTime
	System.String	GetString ¹
TIMESTAMP WITH TIME ZONE	System.DateTimeOffset	GetDateTimeOffset
	System.String	GetString ¹
UROWID	System.String	GetString
	System.Char[]	GetChars
VARCHAR2	System.String	GetString
	System.Char[]	GetChars
	System.Boolean	GetBoolean

Table 3-22 (Cont.) .NET Type Accessors

Oracle Native Data Type	.NET Type	Typed Accessor
VECTOR	System.Byte[]	GetByteArray
	System.Double[]	GetDoubleArray
	System.Single[]	GetFloatArray
	System.Int16[]	GetInt16Array
	System.String	GetString
XMLType	System.String	GetString
	System.Xml.XmlReader	GetXmlReader

- Superscript ¹: The typed accessor is available in managed ODP.NET and ODP.NET Core only. The GetString accessors were added starting in ODP.NET 19.10.
- Superscript ²: The typed accessor is available in unmanaged ODP.NET only.

Certain methods and properties of the `OracleDataReader` object require ODP.NET to map a `NUMBER` column to a .NET type based on the precision and scale of the column. These members are:

- `Item` property
- `GetFieldType` method
- `GetValue` method
- `GetValues` method

ODP.NET determines the appropriate .NET type by considering the following .NET types in order, and selecting the first .NET type from the list that can represent the entire range of values of the column:

- `System.Byte`
- `System.Int16`
- `System.Int32`
- `System.Int64`
- `System.Single`
- `System.Double`
- `System.Decimal`

If no .NET type exists that can represent the entire range of values of the column, then an attempt is made to represent the column values as a `System.Decimal` type. If the value in the column cannot be represented as `System.Decimal`, then an exception is raised.

For example, consider two columns defined as `NUMBER(4,0)` and `NUMBER(10,2)`. The first .NET types from the previous list that can represent the entire range of values of the columns are `System.Int16` and `System.Double`, respectively. However, consider a column defined as `NUMBER(20,10)`. In this case, there is no .NET type that can represent the entire range of values on the column, so an attempt is made to return values in the column as a `System.Decimal` type. If a value in the column cannot be represented as a `System.Decimal` type, then an exception is raised.

The `Fill` method of the `OracleDataAdapter` class uses the `OracleDataReader` object to populate or refresh a `DataTable` or `DataSet` with .NET types. As a result, the .NET type used to

represent a NUMBER column in the DataTable or DataSet also depends on the precision and scale of the column.

See Also

- "[OracleDataReader Class](#)"
- "[OracleDataAdapter Class](#) "
- "[Item](#) "
- "[GetFieldType](#) "
- "[GetValues](#) "
- "[GetValue](#) "

ODP.NET Type Accessors

ODP.NET exposes provider-specific types that natively represent the data types in the database. In some cases, these ODP.NET types provide better performance and functioning than the corresponding .NET types. The ODP.NET types can be obtained from the OracleDataReader object by calling their respective typed accessor.

[Table 3-23](#) lists the valid type accessors that ODP.NET uses to obtain ODP.NET types for an Oracle native type.

Table 3-23 ODP.NET Type Accessors

Oracle Native Data Type	ODP.NET Type	Typed Accessor
BFILE	OracleBFile	GetOracleBFile
BINARY_DOUBLE	OracleDecimal	GetOracleDecimal
BINARY_FLOAT	OracleDecimal	GetOracleDecimal
BLOB	OracleBlob	GetOracleBlob
	OracleBlob	GetOracleBlobForUpdate
	OracleBinary	GetOracleBinary
BOOLEAN	OracleBoolean	GetOracleBoolean
CHAR	OracleString	GetOracleString
CLOB	OracleClob	GetOracleClob
	OracleClob	GetOracleClobForUpdate
	OracleString	GetOracleString
DATE	OracleDate	GetOracleDate
FLOAT	OracleBoolean	GetOracleBoolean
	OracleDecimal	GetOracleDecimal
	OracleString	GetOracleString
INTERVAL DAY TO SECOND	OracleIntervalDS	GetOracleIntervalDS
INTERVAL YEAR TO MONTH	OracleIntervalYM	GetOracleIntervalYM
LONG	OracleString	GetOracleString
LONG RAW	OracleBinary	GetOracleBinary

Table 3-23 (Cont.) ODP.NET Type Accessors

Oracle Native Data Type	ODP.NET Type	Typed Accessor
NCHAR	OracleString	GetOracleString
NCLOB	OracleString	GetOracleString
NUMBER	OracleDecimal	GetOracleDecimal
NVARCHAR2	OracleString	GetOracleString
RAW	OracleBinary	GetOracleBinary
REF	OracleRef	GetOracleRef
REF CURSOR	OracleRefCursor	GetOracleRefCursor
	OracleDataReader	GetOracleDataReader
	OracleString	GetOracleString
ROWID	OracleString	GetOracleString
TIMESTAMP	OracleTimeStamp	GetOracleTimeStamp
TIMESTAMP WITH LOCAL TIME ZONE	OracleTimeStampLTZ	GetOracleTimeStampLTZ
TIMESTAMP WITH TIME ZONE	OracleTimeStampTZ	GetOracleTimeStampTZ
UROWID	OracleString	GetOracleString
VARCHAR2	OracleString	GetOracleString
VECTOR	OracleString	GetOracleString
XMLType	OracleString	GetOracleString
	OracleXmlType	GetOracleXmlType

① See Also

"[ODP.NET Types Overview](#)" for a list of all ODP.NET types

Obtaining LONG and LONG RAW Data

ODP.NET fetches and caches rows from the database during the Read method invocations on the OracleDataReader object. The amount of LONG and LONG RAW column data that is retrieved from this operation is determined by InitialLONGFetchSize. The different behaviors observed when InitialLONGFetchSize is set to 0, greater than 0, and -1 are explained in the following sections.

① Note

ODP.NET does not support the CommandBehavior.SequentialAccess enumeration value. Therefore, LONG and LONG RAW data can be fetched randomly.

Setting InitialLONGFetchSize to Zero or a Value Greater than Zero

The specified amount of `InitialLONGFetchSize` characters or bytes for LONG or LONG RAW column data is retrieved into the cache during the Read method invocations on the `OracleDataReader` object.

By default, `InitialLONGFetchSize` is set to 0. In this case, ODP.NET does not fetch any LONG or LONG RAW column data during the Read method invocations on the `OracleDataReader` object. The LONG or LONG RAW data is fetched when the typed accessor method is explicitly invoked for the LONG or LONG RAW column, which incurs a database round-trip because no data is cached.

If `InitialLONGFetchSize` is set to a value greater than 0, that amount of specified data is cached by ODP.NET during the Read method invocations on the `OracleDataReader` object. If the application requests an amount of data less than or equal to the `InitialLONGFetchSize` through the typed accessor methods, no database round-trip is incurred. However, an additional database round-trip is required to fetch data beyond `InitialLONGFetchSize`.

To obtain data beyond the `InitialLONGFetchSize` characters or bytes, one of the following must be in the select list:

- Primary key
- ROWID
- Unique columns - (defined as a set of columns on which a unique constraint has been defined or a unique index has been created, where at least one of the columns in the set has a NOT NULL constraint defined on it)

To be able to fetch the entire LONG or LONG RAW data without having a primary key column, a ROWID, or unique columns in the select list, set the size of the `InitialLONGFetchSize` property on the `OracleCommand` object to equal or greater than the number of characters or bytes needed to be retrieved.

The LONG or LONG RAW data is returned when the appropriate typed accessor method (`GetChars`, `GetOracleString`, or `GetString` for LONG or `GetOracleBinary` or `GetBytes` for LONG RAW) is called on the `OracleDataReader` object.

Setting InitialLONGFetchSize to -1

By setting `InitialLONGFetchSize` to -1, it is possible to fetch the entire LONG or LONG RAW data from the database for a select query, without requiring a primary key, ROWID, or unique column in the select list.

When `InitialLONGFetchSize` is set to -1, the entire LONG or LONG RAW data is retrieved and cached during Read method invocations on the `OracleDataReader` object. Calls to `GetString`, `GetOracleString`, `GetChars`, `GetBytes`, or `GetOracleBinary` in the `OracleDataReader` return the entire column data.

Obtaining LOB Data

ODP.NET fetches and caches rows from the database during the Read method invocations on the `OracleDataReader` object. The amount of LOB column data that is retrieved from this operation is determined by `InitialLOBFetchSize`.

The following is a complete list of typed accessor methods that an application can call for the CLOB and BLOB columns, if `InitialLOBFetchSize` is set to 0, greater than 0, or -1:

- Methods callable for BLOB column
 - GetBytes
 - GetValue
 - GetValues
 - GetOracleBinary
 - GetOracleBlob
 - GetOracleBlobForUpdate
 - GetOracleValue
 - GetOracleValues
- Methods callable for CLOB column
 - GetChars
 - GetString
 - GetValue
 - GetValues
 - GetOracleString
 - GetOracleClob
 - GetOracleClobForUpdate
 - GetOracleValue
 - GetOracleValues

The following sections explain the different behaviors observed when `InitialLOBFetchSize` is set to 0, greater than 0, and -1.

Setting InitialLOBFetchSize to Zero

By default, the `InitialLOBFetchSize` property is 0. This value dictates to ODP.NET that any LOBs selected will have their client LOB data fetches deferred until after the `OracleDataReader.Read`, such as when using the an accessor. Each LOB value is retrieved only at the point it is individually accessed.

The advantage of using this retrieval strategy is that it conserves client memory and bandwidth. If the LOBs selected are either very large or not necessary to be immediately consumed by the end user, or both, then the application can perform better if LOBs are retrieved as needed, rather than all at once.

Setting InitialLOBFetchSize to a Value Greater than Zero

If `InitialLOBFetchSize` is set to a value greater than 0, ODP.NET caches LOB data up to `InitialLOBFetchSize` characters or bytes for each LOB selected during the `Read` method invocations on the `OracleDataReader` object. The maximum value is 2,147,483,647 (2GB). If the total size of a selected LOB is less than this number, the entire LOB data will be read.

By pre-fetching all LOB entries in one or more database round trips, applications can perform faster by reducing round trips. This approach is most advantageous when most LOBs are either small in size, or consumed by the end user almost immediately, or both. The down side of a large fetch size is higher memory consumption.

This section discusses the ways to fetch beyond the `InitialLOBFetchSize` characters or bytes that are cached.

The remaining LOB data is returned when a typed accessor is invoked, regardless of the value set to the `InitialLOBFetchSize` property. Primary key, ROWID, or unique columns are not required to be in the query select list to obtain data beyond the specified `InitialLOBFetchSize`.

The `GetOracleBlob`, `GetOracleClob`, `GetOracleBlobForUpdate`, and `GetOracleClobForUpdate` methods can now be invoked even if `InitialLOBFetchSize` is greater than 0.

Setting InitialLOBFetchSize to -1

To fetch all LOB data selected during the read operation and not be bound by a set limit per LOB, set `InitialLOBFetchSize` to -1. A new default behavior has been introduced for ODP.NET Release 12.1.0.2 and higher when `InitialLobFetchSize` is set to -1.

When `LegacyEntireLOBFetch = 0`, which is the default value, the following operations are invoked for a LOB column:

- `OracleDataReader.GetOracleClob()`: returns `OracleClob` object
- `OracleDataReader.GetOracleBlob()` : returns `OracleBlob` object
- `OracleDataReader.GetOracleClobForUpdate()`: returns `OracleClob` object
- `OracleDataReader.GetOracleBlobForUpdate()`: returns `OracleBlob` object
- `OracleDataReader.GetOracleValue()`: returns `OracleClob` object for a CLOB column
- `OracleDataReader.GetOracleValue()`: returns `OracleBlob` object for a BLOB column
- `OracleDataAdapter.Fill()` with `ProviderSpecificTypes=true`: populates `DataTable` with `OracleClob` for a CLOB column
- `OracleDataAdapter.Fill()` with `ProviderSpecificTypes=true`: populates `DataTable` with `OracleBlob` for a BLOB column

To use the old behavior, set `LegacyEntireLobFetch = 1` in the ODP.NET configuration.

When `LegacyEntireLobFetch = 1` and `InitialLOBFetchSize = -1`, `GetOracleClob`, `GetOracleClobForUpdate`, `GetOracleBlob`, and `GetOracleBlobForUpdate` methods are not supported. The following operations are invoked for a LOB column in this scenario:

- `OracleDataReader.GetOracleClob()`: throws `InvalidCastException()`
- `OracleDataReader.GetOracleBlob()`: throws `InvalidCastException()`
- `OracleDataReader.GetOracleClobForUpdate()`: throws `InvalidCastException()`
- `OracleDataReader.GetOracleBlobForUpdate()`: throws `InvalidCastException()`
- `OracleDataReader.GetOracleValue()`: returns `OracleString` object for a CLOB column
- `OracleDataReader.GetOracleValue()`: returns `OracleBinary` object for a BLOB column
- `OracleDataAdapter.Fill()` with `ProviderSpecificTypes=true`: populates `DataTable` with `OracleString` for a CLOB column
- `OracleDataAdapter.Fill()` with `ProviderSpecificTypes=true`: populates `DataTable` with `OracleBinary` for a BLOB column

For releases prior to ODP.NET 12.1.0.2, by setting `InitialLOBFetchSize` to -1, it is possible to fetch the entire LOB data from the database for a select query, without requiring a primary key, ROWID, or unique column in the select list. When `InitialLOBFetchSize` is set to -1, the entire

LOB column data is fetched and cached during the Read method invocations on the OracleDataReader object. Calls to GetString, GetOracleString, GetChars, GetBytes, or GetOracleBinary in the OracleDataReader allow retrieving all data.

Methods Supported for InitialLOBFetchSize of -1 and LegacyEntireLobFetch of 1

This section lists supported and not supported methods for the CLOB and BLOB data types when the InitialLOBFetchSize property is set to -1 and LegacyEntireLobFetch property is set to 1.

[Table 3-24](#) lists supported and not supported methods for the CLOB data types.

Table 3-24 Supported OracleDataReader CLOB Methods for InitialLOBFetchSize of -1 and LegacyEntireLobFetch of 1

OracleDataReader CLOB Methods	Supported
GetChars	Yes
GetString	Yes
GetValue	Yes
GetValues	Yes
GetOracleString	Yes
GetOracleValue	Yes
GetOracleValues	Yes
GetOracleClob	No
GetOracleClobForUpdate	No

[Table 3-25](#) lists supported and not supported methods for the BLOB data types.

Table 3-25 Supported OracleDataReader BLOB Methods for InitialLOBFetchSize of -1 and LegacyEntireLobFetch of 1

OracleDataReader BLOB Methods	Supported
GetBytes	Yes
GetValue	Yes
GetValues	Yes
GetOracleBinary	Yes
GetOracleValue	Yes
GetOracleValues	Yes
GetOracleBlob	No
GetOracleBlobForUpdate	No

Performance Considerations Related to the InitialLOBFetchSize Property

This section discusses the advantages and disadvantages of the various InitialLOBFetchSize property settings in different situations.

An application does not have to choose between performance and OracleBlob and OracleClob functionality. Setting the InitialLOBFetchSize property results in a performance boost and still gives the flexibility to use the OracleBlob and OracleClob objects.

If the size of the LOB data is unknown or if the LOB data size varies irregularly, then it is better to leave the `InitialLOBFetchSize` property to its default value of 0. This still gives better performance in most cases.

Setting the `InitialLOBFetchSize` property to a size equal to or greater than the LOB data size for most rows improves performance. It is generally recommended that the `InitialLOBFetchSize` property be set to a value larger than the size of the LOB data for more than 80% of the rows returned by the query. For example, if the size of the LOB data is less than 1 KB in 80% of the rows, and more than 1 MB for 20% of the rows, set the `InitialLOBFetchSize` property to 1 KB.

See Also

- ["LOB Support"](#)
- ["InitialLOBFetchSize"](#)
- ["InitialLONGFetchSize"](#)

Controlling the Number of Rows Fetched in One Database Round-Trip

Application performance depends on the number of rows the application needs to fetch, and the number of database round-trips that are needed to retrieve them. There are two main ways for developers to control the number of rows to fetch per round trip: the `RowsToFetchPerRoundTrip` property or `FetchSize` property.

The easier method is to use `RowsToFetchPerRoundTrip`. On the other hand, `FetchSize` provides more flexibility and granular control if that is needed. In most cases, developers will use `RowsToFetchPerRoundTrip`.

Use of RowsToFetchPerRoundTrip

This `RowsToFetchPerRoundTrip` property represents the total number of rows ODP.NET retrieves from the database per round trip.

If you set the property value to 25, then ODP.NET will retrieve 25 rows in the first data retrieval round trip and another 25 rows in each subsequent data retrieval round trip. If there are fewer rows than 25 that remain, it will retrieve whatever number of rows are left in the result set.

The more rows that are retrieved per round trip, the faster the performance, especially for the result sets that have many, many rows. The downside is that more memory is required to store the data within the larger number of rows.

This property is available in managed ODP.NET and ODP.NET Core only. It can be set on the .NET configuration file, `OracleConfiguration`, `OracleConnection`, `OracleCommand`, `OracleDataReader`, or `OracleRefCursor` objects.

Use of FetchSize

The `FetchSize` property represents the total memory size in bytes that ODP.NET allocates to cache the data fetched from a database round-trip.

The `FetchSize` property can be set on the `OracleCommand`, `OracleDataReader`, or `OracleRefCursor` object, depending on the situation. It controls the fetch size for filling a `DataSet` or `DataTable` using an `OracleDataAdapter`.

If the `FetchSize` property is set on the `OracleCommand` object, then the newly created `OracleDataReader` object inherits the `FetchSize` property of the `OracleCommand` object. This inherited `FetchSize` value can be left as is, or modified to override the inherited value. The `FetchSize` property of the `OracleDataReader` object can be changed before the first `Read` method invocation, which allocates memory specified by the `FetchSize` property. All subsequent fetches from the database use the same cache allocated for that `OracleDataReader` object. Therefore, changing the `FetchSize` value after the first `Read` method invocation has no effect.

Fine-Tuning FetchSize

By fine-tuning the `FetchSize` property, applications can control memory usage and the number of rows fetched in one database round-trip for better performance.

For example, if a query returns 100 rows and each row takes 1024 bytes, then setting the `FetchSize` property to 102400 takes just one database round-trip to fetch 100 rows. For the same query, if the `FetchSize` property is set to 10240, it takes 10 database round-trips to retrieve 100 rows. If the application requires all the rows to be fetched from the result set, the first scenario is faster than the second. However, if the application requires just the first 10 rows from the result set, the second scenario can perform better because it fetches only 10 rows, not 100 rows. When the next 10 rows are fetched, then the memory allocated for rows 1-10 is reused for rows 11-20.

The larger the `FetchSize`, the more system memory is used. Developers should not set large fetch sizes if their client systems have limited memory resources.

Using the RowSize Property

The `RowSize` property of the `OracleCommand` or `OracleRefCursor` object is populated with the row size (in bytes) after an execution of a `SELECT` statement. The `FetchSize` property can then be set to a value relative to the `RowSize` property by setting it to the result of multiplying the `RowSize` value times the number of rows to fetch for each database round-trip.

For example, setting the `FetchSize` to `RowSize * 10` forces the `OracleDataReader` object to fetch exactly 10 rows for each database round-trip. Note that the `RowSize` value does not change due to the data length in each individual column. Instead, the `RowSize` value is determined strictly from the metadata information of the database table(s) that the `SELECT` statement is executed against.

The `RowSize` property can be used to set the `FetchSize` property at design time or at run time, as described in the following sections.

Setting FetchSize Value in the Registry

The `HKLM\Software\Oracle\ODP.NET\ version\FetchSize` registry entry can be set to specify the default result set fetch size (in bytes) for all applications that use that particular version of ODP.NET or the `FetchSize` attribute in the application configuration or `web.config` file can specify the default value for a given application. By default, the fetch size is 131072 bytes. This value can be overridden programmatically by having the applications set the `FetchSize` property on either the `OracleCommand` or the `OracleDataReader` at run time.

Setting FetchSize Value at Design Time

If the row size for a particular `SELECT` statement is already known from a previous execution, the `FetchSize` value of the `OracleCommand` object can be set at design time to the result of multiplying that row size times the number of rows the application wishes to fetch for each

database round-trip. The `FetchSize` value set on the `OracleCommand` object is inherited by the `OracleDataReader` object that is created by the `ExecuteReader` method invocation on the `OracleCommand` object. Rather than setting the `FetchSize` value on the `OracleCommand` object, the `FetchSize` value can also be set on the `OracleDataReader` object directly. In either case, the `FetchSize` value is set at design time, without accessing the `RowSize` property value at run time.

Setting FetchSize Value at Run Time

Applications that do not know the row size at design time can use the `RowSize` property of the `OracleCommand` object to set the `FetchSize` property of the `OracleDataReader` object. The `RowSize` property provides a dynamic way of setting the `FetchSize` property based on the size of a row.

After an `OracleDataReader` object is obtained by invoking the `ExecuteReader` method on the `OracleCommand` object, the `RowSize` property is populated with the size of the row (in bytes). By using the `RowSize` property, the application can dynamically set the `FetchSize` property of the `OracleDataReader` object to the product of the `RowSize` property value multiplied by the number of rows the application wishes to fetch for each database round-trip. In this scenario, the `FetchSize` property is set by accessing the `RowSize` property at run time.

PL/SQL REF CURSOR and OracleRefCursor

The `REF CURSOR` is a data type in the Oracle PL/SQL language. It represents a cursor or a result set in Oracle Database. The `OracleRefCursor` object is a corresponding ODP.NET type for the `REF CURSOR` type.

This section discusses the following aspects of using the `REF CURSOR` data type and `OracleRefCursor` objects:

- [Obtaining an OracleRefCursor Object](#)
- [Obtaining a REF CURSOR Data Type](#)
- [Populating an OracleDataReader from a REF CURSOR](#)
- [Populating the DataSet from a REF CURSOR](#)
- [Populating an OracleRefCursor from a REF CURSOR](#)
- [Updating a DataSet Obtained from a REF CURSOR](#)
- [Behavior of ExecuteScalar Method for REF CURSOR](#)
- [Passing a REF CURSOR to a Stored Procedure](#)

Obtaining an OracleRefCursor Object

There are no constructors for `OracleRefCursor` objects. They can be acquired only as parameter values from PL/SQL stored procedures, stored functions, or anonymous blocks.

Starting with managed ODP.NET and ODP.NET Core 23.5, they can also be acquired by `OracleDataReader.GetOracleRefCursor()` from a `REF CURSOR` type column without any parameters.

An `OracleRefCursor` object is a connected object. The connection used to execute the command returning an `OracleRefCursor` object is required for its lifetime. Once the connection associated with an `OracleRefCursor` object is closed, the `OracleRefCursor` object cannot be used.

Obtaining a REF CURSOR Data Type

A REF CURSOR data type can be obtained as an `OracleDataReader`, `DataSet`, or `OracleRefCursor` object. If the REF CURSOR data type is obtained as an `OracleRefCursor` object, it can be used to create an `OracleDataReader` object or populate a `DataSet` from it. When accessing a REF CURSOR data type, always bind it as an `OracleDbType.RefCursor` parameter.

Starting with managed ODP.NET and ODP.NET Core 23.5, a REF CURSOR type column value can be obtained as an `OracleDataReader`, or `OracleRefCursor` object from `OracleDataReader`; its result set can be a JSON formatted string or `OracleString` from `OracleDataReader`.

Populating an OracleDataReader from a REF CURSOR

A REF CURSOR data type can be obtained as an `OracleDataReader` object by calling the `ExecuteReader` method of the `OracleCommand` object. The output parameter with the `OracleDbType` property set is bound to `OracleDbType.RefCursor`. None of the output parameters of type `OracleDbType.RefCursor` is populated after the `ExecuteReader` method is invoked.

If there are multiple output REF CURSOR parameters, use the `NextResult` method of the `OracleDataReader` object to access the next REF CURSOR data type. The `OracleDataReader.NextResult` method provides sequential access to the REF CURSOR data types; only one REF CURSOR data type can be accessed at a given time.

The order in which `OracleDataReader` objects are created for the corresponding REF CURSOR data types depends on the order in which the parameters are bound. If a PL/SQL stored function returns a REF CURSOR data type, then it becomes the first `OracleDataReader` object and all the output REF CURSOR data types follow the order in which the parameters are bound.

Starting with managed ODP.NET and ODP.NET Core 23.5, a REF CURSOR type column value can be obtained as an `OracleDataReader` object by calling `Read()` and `GetOracleDataReader(int i)` methods of the `OracleDataReader` object using the index of the REF CURSOR type column.

Populating the DataSet from a REF CURSOR

For the `Fill` method to populate the `DataSet` properly, the `SelectCommand` property of the `OracleDataAdapter` class must be bound with an output parameter of type `OracleDbType.RefCursor`. If the `Fill` method is successful, the `DataSet` is populated with a `DataTable` that represents a REF CURSOR data type.

If the command execution returns multiple REF CURSOR data types, the `DataSet` is populated with multiple `DataTable` objects.

With Oracle Data Provider for .NET release 11.1.0.6.20, the extended property, `RefCursorName`, has been introduced on the `DataTable`, to identify the REF CURSOR that populates the `DataTable`.

This property is particularly useful when a `DataSet` is being populated with more than one REF CURSOR, one or more of which is NULL. For example, if a `DataSet` is populated by executing a stored procedure that returns three REF CURSORS and the second REF CURSOR is NULL, the `RefCursorName` property value for the first `DataTable` is `RefCursor` and for the second `DataTable`, `RefCursor2`. No `DataTable` is populated for the NULL REF CURSOR.

Populating an OracleRefCursor from a REF CURSOR

When the `ExecuteNonQuery` method is invoked on a command that returns one or more REF CURSOR data types, each of the `OracleCommand` parameters that are bound as an `OracleDbType.RefCursor` gets a reference to an `OracleRefCursor` object.

To create an `OracleDataReader` object from an `OracleRefCursor` object, invoke the `GetDataReader` method from the `OracleRefCursor` object. Subsequent calls to the `GetDataReader` method return a reference to the same `OracleDataReader` object.

To populate a `DataSet` with an `OracleRefCursor` object, the application can invoke a `Fill` method of the `OracleDataAdapter` class that takes an `OracleRefCursor` object. Similar to the `OracleDataReader` object, an `OracleRefCursor` object is forward-only. Therefore, once a row is read from an `OracleRefCursor` object, that same row cannot be obtained again from it unless it is populated again from a query.

When multiple REF CURSOR data types are returned from a command execution as `OracleRefCursor` objects, the application can choose to create an `OracleDataReader` object or populate a `DataSet` with a particular `OracleRefCursor` object. All the `OracleDataReader` objects or `DataSet` objects created from the `OracleRefCursor` objects are active at the same time, and can be accessed in any order.

Starting with managed ODP.NET and ODP.NET Core 23.5, a REF CURSOR type column value can be obtained as an `OracleRefCursor` object by calling `Read()` and `GetOracleOracleRefCursor(int i)` methods of the `OracleDataReader` object using the index of the REF CURSOR type column.

Updating a DataSet Obtained from a REF CURSOR

REF CURSOR types cannot be updated. However, data that is retrieved into a `DataSet` can be updated. Therefore, the `OracleDataAdapter` class requires a custom SQL statement to flush any REF CURSOR data updates to the database.

The `OracleCommandBuilder` object cannot be used to generate SQL statements for REF CURSOR updates.

Behavior of ExecuteScalar Method for REF CURSOR

The `ExecuteScalar` method returns the value of the first column of the first row of the REF CURSOR if it is one of the following:

- A return value of a stored function execution
- The first bind parameter of a stored procedure execution

See Also

Oracle AI Database SecureFiles and Large Objects Developer's Guide for more information

Passing a REF CURSOR to a Stored Procedure

An application can retrieve a REF CURSOR type from a PL/SQL stored procedure or function and pass it to another stored procedure or function. This feature is useful in scenarios where a stored procedure or a function returns a REF CURSOR type to the .NET application, and based on the application logic, the application passes this REF CURSOR to another stored procedure for processing. Note that if you retrieve the data from a REF CURSOR type in the .NET application, you cannot pass it back to another stored procedure.

The following example demonstrate passing a REF CURSOR:

```
/*
connect scott/tiger@oracle
create table test (col1 number);
insert into test(col1) values (1);
commit;

create or replace package testPkg as type empCur is REF Cursor;
end testPkg;
/

create or replace procedure testSP(param1 IN testPkg.empCur, param2 OUT NUMBER)
as
begin
FETCH param1 into param2;
end;
/
*/

// C#

using System;
using Oracle.DataAccess.Client;
using System.Data;

class InRefCursorParameterSample
{
    static void Main()
    {
        OracleConnection conn = new OracleConnection
            ("User Id=scott; Password=tiger; Data Source=oracle");

        conn.Open(); // Open the connection to the database

        // Command text for getting the REF Cursor as OUT parameter
        String cmdTxt1 = "begin open :1 for select col1 from test; end;";

        // Command text to pass the REF Cursor as IN parameter
        String cmdTxt2 = "begin testSP (:1, :2); end;";

        // Create the command object for executing cmdTxt1 and cmdTxt2
        OracleCommand cmd = new OracleCommand(cmdTxt1, conn);

        // Bind the Ref cursor to the PL/SQL stored procedure
        OracleParameter outRefPrm = cmd.Parameters.Add("outRefPrm",
            OracleDbType.RefCursor, DBNull.Value, ParameterDirection.Output);

        cmd.ExecuteNonQuery(); // Execute the anonymous PL/SQL block
```

```

// Reset the command object to execute another anonymous PL/SQL block
cmd.Parameters.Clear();
cmd.CommandText = cmdTxt2;

// REF Cursor obtained from previous execution is passed to this
// procedure as IN parameter
OracleParameter inRefPrm = cmd.Parameters.Add("inRefPrm",
    OracleDbType.RefCursor, outRefPrm.Value, ParameterDirection.Input);

// Bind another Number parameter to get the REF Cursor column value
OracleParameter outNumPrm = cmd.Parameters.Add("outNumPrm",
    OracleDbType.Int32, DBNull.Value, ParameterDirection.Output);

cmd.ExecuteNonQuery(); //Execute the stored procedure

// Display the out parameter value
Console.WriteLine("out parameter is: " + outNumPrm.Value.ToString());
}
}

```

Implicit REF CURSOR Binding

ODP.NET enables applications to run stored procedures with REF CURSOR parameters without using explicit binding for these parameters in the .NET code. ODP.NET unmanaged and managed drivers support REF CURSOR implicit binding through configuration done in .NET configuration files.

For a read-only result set, such as a REF CURSOR using `OracleDataReader`, REF CURSOR schema information is retrieved automatically.

For some scenarios, such as when updateable REF CURSORS or Entity Framework is used, developers need to define the REF CURSOR schema information so that the application can bind the implicit REF CURSOR. Entity Framework applications use implicit REF CURSOR binding to instantiate complex types from REF CURSOR data. Applications must specify REF CURSOR bind and metadata information in the `app.config`, `web.config`, or `machine.config` .NET configuration file.

The attributes supplied in the .NET configuration file are also used when the application requests for schema information from the `OracleDataReader` object that represents a REF CURSOR. This means that for REF CURSORS that are created using a SELECT from a single table, the application can update that table through the use of `OracleDataAdapter` and `OracleCommandBuilder`.

When using the Entity Framework, function imports can return an implicitly-bound REF CURSOR. The REF CURSOR can be returned as a collection of complex types or entity types. To return a complex type collection, the .NET configuration file needs to define the REF CURSOR bind and metadata information. To return an entity type collection, only the bind information needs to be defined in the .NET configuration file.

This section contains the following topics:

- [Specifying REF CURSOR Bind and Metadata Information in the .NET Configuration File](#)
- [Sample Configuration File and Application](#)
- [Usage Considerations](#)

Specifying REF CURSOR Bind and Metadata Information in the .NET Configuration File

Specify the REF CURSOR information in the `oracle.dataaccess.client` or in the `oracle.manageddataaccess.client` configuration section of the .NET configuration file. Use an `<add>` element for each piece of information. The `add` element uses name-value attributes to specify REF CURSOR information. Use the following format to specify bind information:

```
<add
name="SchemaName.PackageName.StoredProcedureName.RefCursor.RefCursorParameterPositionOrName"
value="implicitRefCursor bindinfo='mode=InputOutput|Output|ReturnValue'" />
```

Use the following format to specify metadata information:

```
<add
name="SchemaName.PackageName.StoredProcedureName.RefCursorMetaData.RefCursorParameterPositionOrName.Column.ColumnOrdinal"
value="implicitRefCursor metadata=AttributesList" />
```

Each REF CURSOR column needs to have an `add` element defined for it. For example, if you have a REF CURSOR returning five columns, then you need to define five `add` elements in the config file.

Each `add` element contains the name and value attributes. The value attribute must begin with the word `implicitRefCursor` followed by the `bindinfo` or `metadata` attribute for specifying bind or metadata information.

The `bindinfo` information is used by ODP.NET for binding REF CURSOR parameters. The `metadata` information is used by ODP.NET to associate the schema information with the appropriate REF CURSOR. The metadata comprises of an attributes list that includes parameters together with their values.

The `SchemaName`, `PackageName`, and `StoredProcedureName` are case-sensitive. In order to run a stored procedure with implicit REF CURSOR binding, the `SchemaName.PackageName.StoredProcedureName` portion of the name attribute must exactly match the name specified in the data dictionary for that stored procedure.

Note

If the application uses implicit REF CURSOR binding feature outside of Entity Framework, then the .NET configuration file and `OracleCommand.CommandText` do not require the schema name concatenated before the stored procedure name.

If any schema, package, or stored procedure name in the database contains lowercase characters, then it must be enclosed within double quotation marks (") in the config file to preserve the case. Double quotation marks are used within the name attribute by using `"`; when needed. For example, if the schema name is `HrSchema`, the package name is `HrPackage`, and the stored procedure name is `HrStoredProcedure` in the database, the config file should use the following:

```
<add
name="&quot;HrSchema&quot;;&quot;HrPackage&quot;;&quot;HrStoredProcedure&quot;;.RefCursorMetadata . . ." />
```

By default, Oracle Database stores these names as uppercase characters. ODP.NET assumes default behavior, and converts all names to uppercase characters unless you explicitly preserve the case by using double quotation marks.

① Note

The *SchemaName*, *PackageName*, *StoredProcedureName*, or *ParameterName* cannot contain a period (".") in the name. For example, P.0 is an unacceptable parameter name.

Depending on whether the application uses bind-by-name or bind-by-position, the *RefCursorParameterPositionOrName* portion of the name attribute must be set with the correct parameter position (for bind by position) or parameter name (for bind by name). For functions, the position is 0-based, where the position 0 represents the return value. For procedures, the position is 1-based, as there are no return values for procedures. For example, if a stored procedure accepts five parameters, returning only two REF CURSORS in the third and fifth parameter positions, then the .NET config REF CURSOR bind information should contain one entry for position 3 and one entry for position 5.

If bind-by-name is used, the attribute name is used to identify the REF CURSOR parameter. The name should use the same name and case as the one specified in the data dictionary for that stored procedure.

For *bindinfo*, the mode specifies the parameter direction of the parameter. The mode must be either *InputOutput*, *Output*, or *ReturnValue*.

① Note

Implicit REF CURSOR binding for an input REF CURSOR parameter is not supported.

An exception is thrown at runtime if the .NET configuration file contains an entry for a REF CURSOR whose mode is set to *Input*.

For metadata, The *AttributesList* contains the list of parameters. [Table 3-26](#) describes the parameters that can be included in the *AttributesList*.

[Example 3-5](#) shows a sample add element that uses *bindinfo*. Here, the schema name is SCOTT and the stored procedure name is TESTPROC. The parameter name is parameter1. The mode is output.

[Example 3-6](#) shows a sample add element that uses metadata.

Table 3-26 Allowed Parameters in Attributes List

Name	Type	Required/Optional for Entity Framework	Description
ColumnName	System.String	Required	The name of the column.

Table 3-26 (Cont.) Allowed Parameters in Attributes List

Name	Type	Required/Optional for Entity Framework	Description
ProviderType	Oracle.DataAccess.Client.OracleDbType or Oracle.ManagedDataAccess.Client.OracleDbType	Required	The database column type (OracleDbType) of the column
NativeDataType	System.String	Required	The Oracle type. For example, NCL0B.
BaseColumnName	System.String	Optional	The name of the column in the database if an alias is used for the column.
BaseSchemaName	System.String	Optional	The name of the schema in the database that contains the column.
BaseTableName	System.String	Optional	The name of the table or view in the database that contains the column.
ColumnSize	System.Int64	Optional	The maximum possible length of a value in the column
NumericPrecision	System.Int16	Optional	The maximum precision of the column, if the column is a numeric data type.
NumericScale	System.Int16	Optional	The maximum scale of the column, if the column is a numeric data type.
IsUnique	System.Boolean	Optional	Indicates whether or not the column is unique.
IsKey	System.Boolean	Optional	Indicates whether or not the column is a key column. For a table to be updated with the REF CURSOR information, at least one of the columns in the REF CURSOR metadata should have this value set to true
IsRowID	System.Boolean	Optional	true if the column is a ROWID, otherwise false.
DataType	System.RuntimeType	Optional	Maps to the common language runtime type.
AllowDBNull	System.Boolean	Optional	true if null values are allowed, otherwise false
IsAliased	System.Boolean	Optional	true if the column is an alias; otherwise false.
IsByteSemantic	System.Boolean	Optional	IsByteSemantic is: - true if the ColumnSize value uses bytes semantics - false if ColumnSize uses character semantics

Table 3-26 (Cont.) Allowed Parameters in Attributes List

Name	Type	Required/Optional for Entity Framework	Description
IsExpression	System.Boolean	Optional	true if the column is an expression, else false.
IsHidden	System.Boolean	Optional	true if the column is hidden, else false.
IsReadOnly	System.Boolean	Optional	true if the column is read-only, else false
IsLong	System.Boolean	Optional	true if the column is of LONG, LONG RAW, BLOB, CLOB, or BFILE type, else false.
UdtTypeName	System.String	Optional	The type name of the UDT.
ProviderDbType	System.Data.DbType	Optional	System.Data.DbType
ObjectName	System.String	Optional	Represents the name of the object.

Some of the attributes, listed in [Table 3-26](#), automatically have their values set using the result set's metadata. Developers can override these default values by setting a value explicitly.

You may have to explicitly define some attributes listed as optional for certain operations. For example, updateable REF CURSOR requires the developer to define key information.

Example 3-5 Using the add Element with bindinfo

```
<add name="SCOTT.TESTPROC.RefCursor.parameter1" value="implicitRefCursor"
  bindinfo='mode=Output' />
```

Example 3-6 Using the add Element with metadata

```
<add name="scott.TestProc.RefCursorMetaData.parameter1.Column.0"
  value="implicitRefCursor metadata='ColumnName=EMPNO;BaseColumnName=EMPNO;
  BaseSchemaName=SCOTT;BaseTableName=EMP;NativeDataType=number;
  ProviderType=Int32;DataType=System.Int32;ColumnSize=4;AllowDBNull=false;
  IsKey=true' />
```

Sample Configuration File and Application

This section builds a sample application to illustrate implicit REF CURSOR binding. It contains the following topics:

- [Sample Stored Procedure and Function](#)
- [Sample Application Configuration File](#)
- [Sample Application That Uses the Configuration File](#)

Sample Stored Procedure and Function

```
CREATE OR REPLACE FUNCTION GETEMP (
  EMPID IN NUMBER) return sys_refcursor is
  emp sys_refcursor;
BEGIN
  OPEN emp FOR SELECT empno, ename FROM emp where empno = EMPID;
```

```

        return emp;
    END;
/

CREATE OR REPLACE PROCEDURE "GetEmpAndDept" (
    EMPS OUT sys_refcursor,
    DEPTS OUT sys_refcursor) AS
BEGIN
    OPEN EMPS for SELECT empno, ename from emp;
    OPEN DEPTS for SELECT deptno, dname from dept;
END;
/

```

Sample Application Configuration File

```

<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <oracle.dataaccess.client>
    <settings>

      <!-- The following is for SCOTT.GETEMP -->
      <add name="SCOTT.GETEMP.RefCursor.0"
        value="implicitRefCursor bindinfo='mode=ReturnValue'" />

      <!-- The following is for SCOTT.GETEMP's REF CURSOR metadata -->
      <add name="SCOTT.GETEMP.RefCursorMetaData.0.Column.0"
        value="implicitRefCursor metadata='ColumnName=EMPNO;
        BaseColumnName=EMPNO;BaseSchemaName=SCOTT;BaseTableName=EMP;
        NativeDataType=number;ProviderType=Int32;ProviderDbType=Int32;
        DataType=System.Int32;ColumnSize=4;NumericPrecision=10;
        NumericScale=3;AllowDBNull=false;IsKey=true'" />

      <add name="SCOTT.GETEMP.RefCursorMetaData.0.Column.1"
        value="implicitRefCursor metadata='ColumnName=ENAME;
        BaseColumnName=ENAME;BaseSchemaName=SCOTT;BaseTableName=EMP;
        NativeDataType=varchar2;ProviderType=Varchar2;
        ProviderDbType=String;DataType=System.String;
        ColumnSize=10;AllowDBNull=true'" />

      <!-- The following is for "SCOTT"."GetEmpAndDept" -->
      <add name="SCOTT."GetEmpAndDept".RefCursor.EMPS"
        value="implicitRefCursor bindinfo='mode=Output'" />

      <!-- The following is for SCOTT.GETEMP's EMPS REF CURSOR metadata -->
      <add name="SCOTT."GetEmpAndDept".RefCursorMetaData.EMPS.Column.0"
        value="implicitRefCursor metadata='ColumnName=EMPNO;
        BaseColumnName=EMPNO;BaseSchemaName=SCOTT;BaseTableName=EMP;
        NativeDataType=number;ProviderType=Int32;ProviderDbType=Int32;
        DataType=System.Int32;ColumnSize=4;NumericPrecision=10;
        NumericScale=3;AllowDBNull=false;IsKey=true'" />

      <add name="SCOTT."GetEmpAndDept".RefCursorMetaData.EMPS.Column.1"
        value="implicitRefCursor metadata='ColumnName=ENAME;
        BaseColumnName=ENAME;BaseSchemaName=SCOTT;BaseTableName=EMP;
        NativeDataType=varchar2;ProviderType=Varchar2;
        ProviderDbType=String;DataType=System.String;
        ColumnSize=10;AllowDBNull=true'" />

      <!-- The following is for SCOTT.GETEMP's DEPTS REF CURSOR metadata -->
      <add name="SCOTT."GetEmpAndDept".RefCursor.DEPTS"

```

```

        value="implicitRefCursor bindinfo='mode=Output'" />

<add name="SCOTT.&quot;GetEmpAndDept&quot;
      .RefCursorMetaData.DEPTS.Column.0"
      value="implicitRefCursor metadata='ColumnName=DEPTNO;
      BaseColumnName=DEPTNO;BaseSchemaName=SCOTT;BaseTableName=DEPT;
      NativeDataType=number;ProviderType=Int32;ProviderDbType=Int32;
      DataType=System.Int32;ColumnSize=4;NumericPrecision=10;
      NumericScale=3;AllowDBNull=false;IsKey=true'" />

<add name="SCOTT.&quot;GetEmpAndDept&quot;
      .RefCursorMetaData.DEPTS.Column.1"
      value="implicitRefCursor metadata='ColumnName=DNAME;
      BaseColumnName=DNAME;BaseSchemaName=SCOTT;BaseTableName=DEPT;
      NativeDataType=varchar2;ProviderType=Varchar2;
      ProviderDbType=String;DataType=System.String;
      ColumnSize=10;AllowDBNull=true'" />
</settings>
</oracle.dataaccess.client>
</configuration>

```

Sample Application That Uses the Configuration File

```

using System;
using System.Data;
using Oracle.DataAccess.Client;

class Program
{
    static void Main(string[] args)
    {
        try
        {
            // Open a connection
            string constr =
                "User Id=scott;Password=tiger;Data Source=inst1";
            OracleConnection con = new OracleConnection(constr);
            con.Open();

            // Use implicit REF CURSOR binding
            // to execute SCOTT.GETEMP function
            // Use bind by position as configured
            // in app.config for SCOTT.GETEMP
            OracleCommand cmd = con.CreateCommand();
            cmd.CommandText = "SCOTT.GETEMP";
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.BindByName = false;
            OracleParameter empid = cmd.Parameters.Add("empid",
                OracleDbType.Int32, ParameterDirection.Input);
            empid.Value = 7654;

            // Populate the DataSet
            OracleDataAdapter adapter = new OracleDataAdapter(cmd);
            DataSet ds = new DataSet();
            adapter.Fill(ds);
            Console.WriteLine("Retrieved {0} row from EMP",
                ds.Tables[0].Rows.Count);

            // Use implicit REF CURSOR binding
            // to execute "SCOTT"."GetEmpAndDept" procedure
            // Use bind by name as configured
            // in app.config for "SCOTT"."GetEmpAndDept"

```



```

        cmd = con.CreateCommand();
        cmd.CommandText = "\"SCOTT\".\"GetEmpAndDept\"";
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.BindByName = true;
        adapter = new OracleDataAdapter(cmd);
        adapter.Fill(ds);
        Console.WriteLine("Retrieved {0} rows from DEPT",
            ds.Tables[1].Rows.Count);
    }
    catch (Exception ex)
    {
        // Output the message
        Console.WriteLine(ex.Message);
        if (ex.InnerException != null)
        {
            // If any details are available regarding
            // errors in the app.config, print them out
            Console.WriteLine(ex.InnerException.Message);
            if (ex.InnerException.InnerException != null)
            {
                Console.WriteLine(
                    ex.InnerException.InnerException.Message);
            }
        }
    }
}
}
}

```

Usage Considerations

This section discusses the following usage considerations when using implicit REF CURSOR:

- [CommandText Property Considerations](#)
- [Bind Considerations](#)
- [Overloaded Stored Procedures](#)
- [Type Initialization Exceptions](#)
- [Using Stored Functions with Function Import](#)

CommandText Property Considerations

ODP.NET applications should ensure that the stored procedure name and the `OracleCommand.CommandText` match exactly. Let's take a scenario where the stored procedure name in the database is `SCOTT.TESTPROC`. Now, if the `CommandText` uses `TESTPROC`, ODP.NET will look for entries matching `TESTPROC` only. The current schema name will not be automatically appended to `TESTPROC`. So, the correct `CommandText` to use in this scenario would be `SCOTT.TESTPROC`.

Also, the `CommandText` is case-sensitive and must use the same case as the stored procedure name in the database. So if the stored procedure name in the database is `SCOTT.Testproc`, then the `CommandText` must use `SCOTT.Testproc`.

Bind Considerations

If information about a REF CURSOR parameter has been added to the configuration file, then applications should not try to explicitly bind the REF CURSOR parameter to `OracleCommand`. ODP.NET automatically binds the REF CURSOR parameter at the appropriate locations based on the information provided in the configuration file. If the application stored procedure also has

non-REF CURSOR parameters, then these parameters must still be explicitly bound to `OracleCommand`.

If the information specified in the configuration file for a stored procedure identifies the REF CURSOR parameter by name, then all the other non-REF CURSOR parameters should also be bound by name. Also the `BindByName` property for the `OracleCommand` object should be set to `true` in this case. Entity Framework always uses `BindByName` to run stored procedures. Your .NET configuration file parameter names must use the same case that was used when creating the stored procedure in the database.

If the `OracleCommand.BindByName` property is set to `false` (default), then ODP.NET assumes that the parameters have been bound based on their position, and all parameters have been specified in the correct order. For such cases, the parameters specified in the configuration file are bound in the same order in which they appear in the configuration file.

Overloaded Stored Procedures

ODP.NET does not support multiple stored procedures with the same name inside the configuration file. If an ODP.NET application uses an overloaded stored procedure, the application can store only one overloaded stored procedure information in the configuration file.

Type Initialization Exceptions

Type initialization exceptions can be caused by invalid .NET configuration file entries. Evaluate the exception that is caught as well as its inner exceptions to determine the .NET configuration file entry or the attribute setting that is causing the exception.

ODP.NET tracing logs the valid and invalid .NET configuration file entries that ODP.NET has parsed. To look for .NET configuration file related entries, set the `TraceLevel` to the *Entry, exit, and SQL statement information* level setting. Trace entries related to implicit REF CURSOR binding have a (REFCURSOR) entry along with (ERROR), if any errors are encountered.

Using Stored Functions with Function Import

Function Import only supports stored procedures, and does not support functions. When using the **Add Function Import** dialog for the Entity Data Model that you have created, the **Get Column Information** button does not return the metadata information for the REF CURSOR that is being returned by a stored function, even if it is configured properly in the .NET configuration file.

LOB Support

ODP.NET provides an easy and optimal way to access and manipulate large object (LOB) data types.

Note

SecureFiles can be used with existing ODP.NET LOB classes.

This section includes the following topics:

- [Large Character and Large Binary Data Types](#)

- [Oracle Data Provider for .NET LOB Objects](#)
- [Updating LOBs Using a DataSet](#)
- [Updating LOBs Using OracleCommand and OracleParameter](#)
- [Updating LOBs Using ODP.NET LOB Objects](#)
- [Temporary LOBs](#)

Large Character and Large Binary Data Types

Oracle Database supports large character and large binary data types.

Large Character Data Types

- CLOB - Character data can store up to 4 gigabytes.
- NCLOB - Unicode National character set data can store up to 4 gigabytes.

Large Binary Data Types

- BLOB - Unstructured binary data can store up to 4 gigabytes.
- BFILE - Binary data stored in external file can store up to 4 gigabytes.

Note

LONG and LONG RAW data types are made available for backward compatibility in Oracle9i, but should not be used in new applications.

Oracle Data Provider for .NET LOB Objects

ODP.NET provides three objects for manipulating LOB data: `OracleBFile`, `OracleBlob`, and `OracleClob`.

[Table 3-27](#) shows the proper ODP.NET object to use for a particular Oracle LOB type.

Table 3-27 ODP.NET LOB Objects

Oracle LOB Type	ODP.NET LOB Object
BFILE	<code>OracleBFile</code>
BLOB	<code>OracleBlob</code>
CLOB	<code>OracleClob</code>
NCLOB	<code>OracleClob</code>

The ODP.NET LOB objects can be obtained by calling the proper typed accessor on the `OracleDataReader` object, or by calling the proper typed accessor as an output parameter on a command execution with the proper bind type.

All ODP.NET LOB objects inherit from the .NET `Stream` class to provide generic `Stream` operations. The LOB data (except for BFILE types) can be updated using the ODP.NET LOB objects by using methods such as `Write`. Data is not cached in the LOB objects when read and write operations are carried out. Therefore, each read or write request incurs a database round-trip. The `OracleClob` object overloads the `Read` method, providing two ways to read data

from a CLOB. The Read method that takes a byte[] as the buffer populates it with CLOB data as Unicode byte array. The Read method that takes a char[] as the buffer populates it with Unicode characters.

Additional methods can also be found on the OracleBFile object. An OracleBFile object must be explicitly opened using the OpenFile method before any data can be read from it. To close a previously opened BFILE, use the CloseFile method.

Every ODP.NET LOB object is a connected object and requires a connection during its lifetime. If the connection associated with a LOB object is closed, then the LOB object is not usable and should be disposed of.

If an ODP.NET LOB object is obtained from an OracleDataReader object through a typed accessor, then its Connection property is set with a reference to the same OracleConnection object used by the OracleDataReader object. If a LOB object is obtained as an output parameter, then its Connection property is set with a reference to the same OracleConnection property used by the OracleCommand object. If a LOB object is obtained by invoking an ODP.NET LOB object constructor to create a temporary LOB, the Connection property is set with a reference to the OracleConnection object provided in the constructor.

The ODP.NET LOB object Connection property is read-only and cannot be changed during its lifetime. In addition, the ODP.NET LOB types object can be used only within the context of the same OracleConnection referenced by the ODP.NET LOB object. For example, the ODP.NET LOB Connection property must reference the same connection as the OracleCommand object if the ODP.NET LOB object is a parameter of the OracleCommand. If that is not the case, ODP.NET raises an exception when the command is executed.

① See Also

Oracle AI Database SecureFiles and Large Objects Developer's Guide for complete information about Oracle Database 10g LOBs and how to use them

Updating LOBs Using a DataSet

BFILE and BLOB data are stored in the DataSet as byte arrays while CLOB and NCLOB data are stored as strings. In a similar manner to other types, an OracleDataAdapter object can be used to fill and update LOB data changes along with the use of the OracleCommandBuilder object for automatically generating SQL.

Note that an Oracle LOB column can store up to 4 GB of data. When the LOB data is fetched into the DataSet, the actual amount of LOB data the DataSet can hold for a LOB column is limited to the maximum size of a .NET string type, which is 2 GB. Therefore, when fetching LOB data that is greater than 2 GB, ODP.NET LOB objects must be used to avoid any data loss.

Updating LOBs Using OracleCommand and OracleParameter

To update LOB columns, LOB data can be bound as a parameter for SQL statements, anonymous PL/SQL blocks, or stored procedures. The parameter value can be set as a .NET Framework type, ODP.NET type, or as an ODP.NET LOB object type. For example, when inserting .NET string data into a LOB column in an Oracle9i database or later, that parameter can be bound as OracleDbType.VarChar2. For a parameter whose value is set to an OracleClob object, the parameter should be bound as OracleDbType.Clob.

Updating LOBs Using ODP.NET LOB Objects

Oracle BFILEs cannot be updated; therefore, `OracleBFile` objects do not allow updates to BFILE columns.

Two requirements must be met to update LOB data using ODP.NET LOB objects:

1. A transaction must be started before a LOB column is selected.

The transaction must be started using the `BeginTransaction` method on the `OracleConnection` object before the command execution, so that the lock can be released when the `OracleTransaction Commit` or `Rollback` method is invoked.

2. The row in which the LOB column resides must be locked; as part of an entire result set, or on a row-by-row basis.

- a. Locking the entire result set

Add the `FOR UPDATE` clause to the end of the `SELECT` statement. After execution of the command, the entire result set is locked.

- b. Locking the row - there are two options:

- Invoke one of the `OracleDataReader` typed accessors (`GetOracleClobForUpdate` or `GetOracleBlobForUpdate`) on the `OracleDataReader` object to obtain an ODP.NET LOB object, while also locking the current row.

This approach requires a primary key, unique column(s), or a ROWID in the result set because the `OracleDataReader` object must uniquely identify the row to re-select it for locking.

- Execute an `INSERT` or an `UPDATE` statement that returns a LOB in the `RETURNING` clause.

Temporary LOBs

Temporary LOBs can be instantiated for `BLOB`, `CLOB`, and `NCLOB` objects. To instantiate an ODP.NET LOB object that represents a temporary LOB, the `OracleClob` or the `OracleBlob` constructor can be used.

Temporary ODP.NET LOB objects can be used for the following purposes:

- To initialize and populate a LOB column with empty or non-empty LOB data.
- To pass a LOB type as an input parameter to a SQL statement, an anonymous PL/SQL block, or a stored procedure.
- To act as the source or the destination of data transfer between two LOB objects as in the `CopyTo` operation.

Note

Temporary LOBs are not transaction aware. Commit and rollback operations do not affect the data referenced by a temporary LOB.

Native JSON Support

Oracle Database 21c adds a native JavaScript Object Notation (JSON) data type. ODP.NET Core, managed, and unmanaged drivers support this native JSON data type starting with version 21.

In ODP.NET, the database JSON data type can be retrieved or passed to the database. When using a .NET string or `OracleString`, it can be bound as a parameter using the `OracleDbType.Json` enumeration value. This enumeration value directs ODP.NET to perform decoding from and encoding to the native Oracle Database JSON binary format, OSON, on the client side, offloading the task from the server side.

Alternatively, JSON data can be bound as parameters to other ODP.NET and .NET data types or not use the `OracleDbType.Json` enumeration value. Oracle Database will then implicitly encode to and decode from OSON to the desired data type format instead of the client in these cases. In `DataSet`, the Oracle JSON type is converted to and stored as either a .NET string or `OracleString`.

Managed ODP.NET and ODP.NET Core JSON features require the `System.Text.Json` assembly be included as a project dependency. ODP.NET does not add the `System.Text.Json` package as a dependency itself. In many cases, .NET Core 3.1 and higher does automatically include this assembly with the .NET runtime, while .NET Framework 4.8 does not.

Unmanaged ODP.NET does not have a requirement for `System.Text.Json`.

JSON Numeric Values

For managed ODP.NET and ODP.NET Core, JSON documents bound as `OracleDbType.Json` input parameters have a 28 precision upper limit for numeric values. In all other cases, up to 38 precision will be retained for JSON numeric values when sent to or retrieved from the database.

Unmanaged ODP.NET supports JSON numeric values with a maximum precision of 38, as does the database.

If a JSON document bound as `OracleDbType.Json` contains numeric values with higher precision than can be retained, then ODP.NET will round the value to the maximum supported precision. If more precision must be retained than the maximum, then store the numeric value as a string by placing double quotes around the JSON value before binding the JSON document as an input parameter.

JSON Relational Duality

JSON Relational Duality is a landmark capability introduced in Oracle Database that provides game-changing flexibility and simplicity for Oracle Database developers. This breakthrough innovation overcomes the historical challenges that developers faced when building applications, either when using the relational model or when using the document model.

JSON Relational Duality delivers a solution with the benefits of both relational tables and JSON documents, without the tradeoffs of either model.

In the database, JSON Relational Duality manifests as fully updatable JSON views over relational data. These views are called JSON Relational Duality Views.

① See Also

To learn more about the feature, refer to the Oracle Database JSON-Relational Duality Developer's Guide

ODP.NET and JSON Relational Duality Views

All ODP.NET provider types (core, managed, and unmanaged) support using JSON Relational Duality Views. ODP.NET supports the feature starting with release 19c.

With 21c and higher, ODP.NET can retrieve this JSON data as a .NET string or `OracleString` data type when parameters are bound with the `OracleDbType.Json` enumeration value.

ODP.NET 19c uses an earlier implementation that does not include this enumeration value. It fetches JSON data as a BLOB data type, then converts it to a string in .NET. With JSON Relational Duality, ODP.NET retrieves the data with `OracleDataReader.GetValue(s)` or `GetOracleValue(s)` methods, but not `GetString` nor `GetOracleString`.

For the ODP.NET developer, the data type conversion and changes are transparent. No .NET data management needs to occur beyond treating the data as a string consisting of JSON in .NET.

Oracle JSON data can also be consumed and operated in .NET `DataSet`. When saving changes back to the Oracle Database from `DataSet`, the default generated `OracleCommandBuilder` insert operations will successfully complete. However, the default updates and deletes require some custom SQL with parameter binds to make these JSON data changes. Here's a code snippet of how to perform these JSON updates and deletes.

```
// "JRDVIEW" is the JSON Relational Duality View being updated in the example below
// Custom UPDATE SQL
string customUpdateSQL = @"UPDATE JRDVIEW SET data = :updatedJSON where json_value(data,
'$.ID.number()') = json_value(:DBData, '$.ID.number()')";
adapter.UpdateCommand = new OracleCommand(customUpdateSQL, conn);
// Parameter bound to the updated data
adapter.UpdateCommand.Parameters.Add("updatedJSON", OracleDbType.Json, 100, "data");
// Parameter bound to row on DB
adapter.UpdateCommand.Parameters.Add("DBData", OracleDbType.Varchar2, 100,
"data").SourceVersion = DataRowVersion.Original;

// Custom DELETE SQL
string customDeleteSQL = @"DELETE FROM JRDVIEW WHERE json_value(data, '$.ID.number()') =
json_value(:DBData, '$.ID.number()')";
adapter.DeleteCommand = new OracleCommand(customDeleteSQL, conn);
// Parameter bound to row on DB
adapter.DeleteCommand.Parameters.Add("DBData", OracleDbType.Varchar2, 100,
"data").SourceVersion = DataRowVersion.Original;

adapter.Update(ds);
```

In the sample code above, the ID field is a numeric type. We can specify it is a number by appending `.number()` to it in the SQL. Doing so is optional. However, the main benefit it provides is it allows the query to leverage indexes on the base tables, thereby improving execution performance. Other data types can be specified:

- `.string()` for string data types
- `.date()` for date data type

- `.timestamp()` for timestamp data type

Note

When modifying the `DataSet`, do not change the `_metadata` member nor any of its constituent parts. That metadata is used to protect against dirty writes back to the database.

Artificial Intelligence Vectors and Semantic Search

Oracle AI Database 26ai introduces semantic search capabilities using Artificial Intelligence (AI) vector search. These capabilities include a new vector data type, vector indexes, and vector search SQL operators that allow the database to store semantic document content, images, and other unstructured data as vectors and then run fast performing similarity queries. The key innovation is that the database better understands user intent and the search context to find similar matches, rather than only find exact matches.

Building upon this capability, the Oracle database can use Generative AI constructs, such as Retrieval Augmented Generation (RAG) to combine large language models (LLMs) and private business data to respond to natural language questions. RAG provides higher accuracy and avoids exposing any of the private data to the LLM training data.

Oracle Database Vector Data Types

There are two Oracle database vector types: dense and sparse. Dense vectors were the first AI vector type Oracle introduced in Oracle AI Database 26ai. They are ideal for data sets in which nearly every dimension stores a non-zero value, meaning they hold much rich and complex information, patterns, and relationships. A great use case for dense vectors is semantic search when word meaning nuances better identify matches.

On the other hand, sparse vectors typically have a large number of dimensions and few non-zero dimension values. To improve efficiency, they only store non-zero values. That makes sparse vectors more storage and memory efficient. Moreover, ODP.NET can process them faster than dense vectors assuming the data profile has few non-zero dimension values. They are suited for other AI application types, such as text retrieval systems in which a search phrase identifies relevant matching documents.

ODP.NET Vectors

Starting with version 23.3.2, managed ODP.NET and ODP.NET Core support Oracle database vector data access as existing .NET or ODP.NET data types: arrays, strings, CLOBs, and VARCHAR2s. In managed ODP.NET and ODP.NET Core 23.8, new ODP.NET vector-specific types for dense vectors (`OracleVector`) and sparse vectors (`OracleSparseVector`) are introduced for richer user interfaces that more naturally and easily operate on vectors.

ODP.NET vector data types are flexible to use. They have one or more dimensions, up to 65,536 (64K) dimensions. Developers can define vectors with a fixed or variable number of dimensions. The numeric format of each dimension in a vector can be of the same numeric format or a different numeric format for each dimension:

Table 3-28 Dense Vector Numeric Formats

Vector Numeric Format	Description	Mapped .NET Type	Mapped ODP.NET Type	Size	Examples
BINARY	packed UINT8 bytes where each dimension is a single bit	Byte	OracleVector	1 byte	Vector(1024, BINARY), Vector(1024, BINARY, DENSE)
INT8	8-bit integers	Int16	OracleVector	2 bytes	Vector(4, INT8), Vector(4, INT8, DENSE)
FLOAT32	32-bit floating point	Float	OracleVector	4 bytes	Vector(768, FLOAT32), Vector(768, FLOAT32, DENSE)
FLOAT64	64-bit floating point	Double	OracleVector	8 bytes	Vector(10000, FLOAT64), Vector(10000, FLOAT64, DENSE)

For example, a one dimension INT8 vector would be Vector(1, INT8). A four dimension FLOAT32 vector would be Vector(4, FLOAT32). A variable dimension FLOAT64 vector would be Vector(*, FLOAT64).

To specify that each vector dimension can be either an INT8, FLOAT32, or FLOAT64 data type, use the * notation, such as Vector(4, *) to represent a four dimensional vector. You may specify a vector as Vector(*, *) to have a variable number of dimensions and variable data type vector.

Table 3-29 Sparse Vector Numeric Formats

Vector Numeric Format	Description	Mapped .NET Type	Mapped ODP.NET Type	Size	Example
BINARY	packed UINT8 bytes where each dimension is a single bit	Byte	OracleSparse Vector	1 byte	Vector(16, BINARY, SPARSE)
INT8	8-bit integers	Int16	OracleSparse Vector	2 bytes	Vector(4, INT8, SPARSE)
FLOAT32	32-bit integers	Float	OracleSparse Vector	4 bytes	Vector(768, FLOAT32, SPARSE)
FLOAT64	64-bit integers	Double	OracleSparse Vector	8 bytes	Vector(10000, FLOAT64, SPARSE)

ODP.NET `OracleVector` (dense) and `OracleSparseVector` (sparse) both can use all four numeric formats.

ODP.NET can retrieve the database vector data type in .NET as any of the following:

Data Type	ODP.NET or .NET Data Type	Description
.NET numeric array	.NET	This array type can be <code>Int16[]</code> , <code>float[]</code> , or <code>double[]</code> depending on the vector's numeric format.
.NET byte array	.NET	Oracle <code>BINARY</code> numeric format must be retrieved as a .NET byte array.

Note

InvalidCastException is thrown for any `Get*Array()` methods, except `GetByteArray()` method, for `BINARY` format.

.NET string	.NET	This contains the vector representation in JSON format.
<code>OracleString</code>	ODP.NET	This contains the vector representation in JSON format.
<code>OracleClob</code>	ODP.NET	This contains the vector representation in JSON format.
<code>OracleVector</code>	ODP.NET	ODP.NET native dense vector data type. This is the base type for all ODP.NET native vector data types.

Data Type	ODP.NET or .NET Data Type	Description
OracleSparseVector	ODP.NET	ODP.NET native sparse vector data type.

ODP.NET can bind vector data as the following .NET types for SQL or stored procedure execution:

Data Type	.NET or ODP.NET Data Type	Bind Type	Description
OracleVector, OracleSparseVector	ODP.NET	OracleDbType.Vector	ODP.NET native vector data types. ODP.NET will bind the Vector data type as the OracleVector's ProviderType.
OracleVector, OracleSparseVector	ODP.NET	OracleDbType.Vector _Binary, OracleDbType.Vector _Int8, OracleDbType.Vector _Float32, OracleDbType.Vector _Float64	ODP.NET native vector data types. ODP.NET will bind the Vector data type based on the binding type accordingly.
Any numeric array	.NET	OracleDbType.Vector	This array type can be any numeric array type. ODP.NET will infer the numeric array type and bind the Vector data type as OracleDbType.Vector _Int8, OracleDbType.Vector _Float32, or OracleDbType.Vector _Float64 accordingly.
Any numeric array	.NET	OracleDbType.Vector _Int8, OracleDbType.Vector _Float32, OracleDbType.Vector _Float64	This array type can be any numeric array type. ODP.NET will bind the Vector data type based on the binding type accordingly.
.NET byte array	.NET	OracleDbType.Vector , OracleDbType.Vector _Binary, OracleDbType.Vector _Int8, OracleDbType.Vector _Float32, OracleDbType.Vector _Float64	All vector numeric formats can be bound as .NET byte arrays. BINARY vectors can only be bound as .NET byte arrays.

Data Type	.NET or ODP.NET Data Type	Bind Type	Description
.NET string, OracleString	.NET or ODP.NET	OracleDbType.Vector	This contains the vector representation in JSON format. If the vector string is in BINARY sparse vector format, ODP.NET will bind the vector data type as OracleDbType.Vector_Binary; otherwise, ODP.NET will bind the vector data type as OracleDbType.Vector_Float32.
.NET string, OracleString	.NET or ODP.NET	OracleDbType.Vector_Binary OracleDbType.Vector_Int8, OracleDbType.Vector_Float32, OracleDbType.Vector_Float64	This contains the vector representation in JSON format. ODP.NET will bind the Vector data type based on the binding type accordingly.
OracleClob	ODP.NET	OracleDbType.Clob OracleDbType.VarChar2	This contains the vector representation in JSON format.

When using vectors with `OracleDataAdapter` `ReturnProviderSpecificTypes` set to `false`, for dense VECTORS, a .NET numeric array will populate the `DataTable` or `DataSet` after the `Fill` method is called; and for sparse VECTORS, a .NET string representing the sparse vector will be used. When `true`, `OracleVector` or `OracleSparseVector` will be used.

For dense VECTOR columns of '*' numeric format and `ReturnProviderSpecificTypes` set to `false`, a mix of `short[]`, `float[]`, `double[]`, and/or `byte[]` will be populated based on the VECTOR's numeric format upon an `OracleDataAdapter.Fill()`. For dense VECTOR columns of all other formats, `short[]`, `float[]`, `double[]`, or `byte[]` will be populated based on the VECTOR column's numeric format.

Since vectors cannot be compared directly with each other, they cannot be used as JOIN keys, ORDER BY keys, GROUP BY keys, or other related scenarios. Thus, for `OracleCommandBuilder`, vector comparison is not included in the SQL WHERE clause of the generated commands.

Vectors as CLOB or VARCHAR2 Data Type

Unmanaged ODP.NET, as well as managed ODP.NET and ODP.NET Core prior to version 23.3.2, support vectors using existing CLOB or VARCHAR2 data types. In these cases, vectors are stored with a JSON format. Data retrieval and manipulation use existing CLOB or VARCHAR2 or .NET APIs.

In later managed ODP.NET and ODP.NET Core versions, vectors can remain stored as CLOBs or VARCHAR2s for backwards compatibility purposes by setting the `OracleConfiguration` `MapVectorColumnAsClob` property to `true`.

When sparse vectors are sent to ODP.NET versions that do not support sparse vectors natively, the vector data becomes CLOB data in JSON dense vector format.

ODP.NET Vector Sample Code

//This ODP.NET artificial intelligence (AI) vector sample demonstrates how to insert, retrieve, update, and delete multi-dimensional FLOAT64, FLOAT32, and INT8 vector data types using the Oracle database.

//Requires ODP.NET 23ai (23.3.2) or higher and Oracle AI Database 26ai (23.4) or higher.

//Add User Id, Password, and Data Source, such as Easy Connect Plus or TNS, to the connection string to connect to the DB.

```
using Oracle.ManagedDataAccess.Client;
```

```
namespace VectorDemo
```

```
{
```

```
    public class VectorDemo
```

```
    {
```

```
        //Provide User Id, Password, and Data Source values for your database.
```

```
        public const string conStr = "User Id=<USER>;Password=<PASSWORD>;Data Source=<DATA SOURCE>;";
```

```
        public const int id = 1;
```

```
        public static void Main(string[] args)
```

```
        {
```

```
            using (OracleConnection con = new OracleConnection(conStr))
```

```
            {
```

```
                using (OracleCommand cmd = con.CreateCommand())
```

```
                {
```

```
                    try
```

```
                    {
```

```
                        con.Open();
```

```
                        cmd.CommandText = "begin " +
```

```
                            "execute immediate 'drop table VectorTable';" +
```

```
                            "exception when others then if sqlcode <> -942
```

```
then raise;" +
```

```
                            "end if;" +
```

```
                            "end;";
```

```
                        cmd.ExecuteNonQuery();
```

```
                            //Create table with 2 dimensional FLOAT64, 3
```

```
dimensional FLOAT32, and 4 dimensional INT8 vector columns
```

```
                            cmd.CommandText = "create table VectorTable (id
```

```
number, float64s vector(2, float64),
```

```
float32s vector(3, float32), int8s vector(4, INT8), constraint pk primary key
```

```
(id));"
```

```
                            cmd.ExecuteNonQuery();
```

```
                            //Insert vector row into DB
```

```
                            InsertVectors();
```

```

        //Update vector values in DB
        UpdateVectors();

        //Delete vector values in DB
        DeleteVectors();
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}

}

}

public static void InsertVectors()
{
    string sql = "insert into VectorTable values
(:id, :float64, :float32, :int8)";
    try
    {
        double[] doubles = new double[] { 1.234, 2.345 };
        float[] floats = new float[] { 1.23f, 2.34f, 3.45f };
        short[] int16s = new short[] { 1, 2, 3, 4 };

        OracleConnection con = new OracleConnection(conStr);
        con.Open();

        OracleCommand cmd = new OracleCommand(sql, con);
        cmd.Parameters.Add("id", OracleDbType.Int16, 0, id,
System.Data.ParameterDirection.Input);

        //Binding FLOAT64 variable as vector - doubles
        cmd.Parameters.Add("float64", OracleDbType.Vector, 0,
doubles, System.Data.ParameterDirection.Input);
        //Binding FLOAT32 variable as vector - floats
        cmd.Parameters.Add("float32", OracleDbType.Vector, 0, floats,
System.Data.ParameterDirection.Input);
        //Binding INT8 variable as vector - int16s
        cmd.Parameters.Add("int8", OracleDbType.Vector, 0, int16s,
System.Data.ParameterDirection.Input);

        // Insert vectors into VectorTable
        cmd.ExecuteNonQuery();

        //Retrieve vector values from DB
        RetrieveVectors();
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}

public static void UpdateVectors()
{

```

```

        string sql = "update VectorTable set float64s=:float64,
float32s=:float32, int8s=:int8 where id=:id";
        try
        {
            double[] doubles = new double[] { 9.876, 8.765 };
            float[] floats = new float[] { 9.87f, 8.76f, 7.65f };
            short[] int16s = new short[] { 9, 8, 7, 6 };

            OracleConnection con = new OracleConnection(conStr);
            con.Open();

            OracleCommand cmd = new OracleCommand(sql, con);

            //Binding FLOAT64 variable as vector - doubles
            cmd.Parameters.Add("float64", OracleDbType.Vector_Float64, 0,
doubles, System.Data.ParameterDirection.Input);
            //Binding FLOAT32 variable as vector - floats
            cmd.Parameters.Add("float32", OracleDbType.Vector_Float32, 0,
floats, System.Data.ParameterDirection.Input);
            //Binding INT8 variable as vector - int16s
            cmd.Parameters.Add("int8", OracleDbType.Vector_Int8, 0,
int16s, System.Data.ParameterDirection.Input);

            cmd.Parameters.Add("id", OracleDbType.Int16, 0, id,
System.Data.ParameterDirection.Input);

            // Update vectors in VectorTable
            cmd.ExecuteNonQuery();

            Console.WriteLine("Database vector values updated!");
            //Retrieve vector values from DB
            RetrieveVectors();
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
    }

    public static void DeleteVectors()
    {
        string sql = "delete from VectorTable where id=:id";
        try
        {
            OracleConnection con = new OracleConnection(conStr);
            con.Open();

            OracleCommand cmd = new OracleCommand(sql, con);
            cmd.Parameters.Add("id", OracleDbType.Int16, 0, id,
System.Data.ParameterDirection.Input);

            // Delete row with vector values from VectorTable
            cmd.ExecuteNonQuery();

            Console.WriteLine("Database vector values deleted!");
            //Confirm vectors removed

```

```

        RetrieveVectors();
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}

public static void RetrieveVectors()
{
    string sql = "select * from VectorTable";
    try
    {
        OracleConnection con = new OracleConnection(conStr);
        con.Open();

        OracleCommand cmd = new OracleCommand(sql, con);
        OracleDataReader reader = cmd.ExecuteReader();

        //Use ODP.NET vector accessors to retrieve data
        if (reader.Read())
        {
            Console.WriteLine("Retrieve FLOAT64S Vector value:");
            double[] vecD = reader.GetDoubleArray(1);
            PrintDoubles(vecD);

            Console.WriteLine("Retrieve FLOAT32S Vector value:");
            float[] vecF = reader.GetFloatArray("FLOAT32S");
            PrintFloats(vecF);

            Console.WriteLine("Retrieve INT8S vector value:");
            short[] vecInt8s = reader.GetInt16Array(3);
            PrintInt16s(vecInt8s);

            Console.WriteLine();
        }
        //Return no results if no vector row is found
        else
            Console.WriteLine("No vector row found.");
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}

static public void PrintDoubles(double[] doubles)
{
    Console.WriteLine("{");
    for (int i = 0; i < doubles.Length - 1; i++)
        Console.WriteLine(doubles[i].ToString() + ", ");
    Console.WriteLine(doubles[doubles.Length - 1].ToString() + "}");
}

static public void PrintFloats(float[] floats)
{
    Console.WriteLine("{");

```



```

        for (int i = 0; i < floats.Length - 1; i++)
            Console.Write(floats[i].ToString() + ", ");
        Console.WriteLine(floats[floats.Length - 1].ToString() + "}");
    }

    static public void PrintInt16s(Int16[] int16s)
    {
        Console.Write("{");
        for (int i = 0; i < int16s.Length - 1; i++)
            Console.Write(int16s[i].ToString() + ", ");
        Console.WriteLine(int16s[int16s.Length - 1].ToString() + "}");
    }
}
}
}

```

.NET Vectors and Semantic Kernel

ODP.NET Core and managed ODP.NET support .NET vectors and Microsoft Semantic Kernel to build AI and vector-powered Oracle Database applications. Oracle.VectorData NuGet package provides these capabilities.

Refer to the following documentation resources describing the features and APIs:

- [Using the Oracle Database Vector Store connector \(Preview\)](#)
- [Oracle Database Vector Store Connector Classes](#)

ODP.NET XML Support

ODP.NET allows the extraction of data from relational and object-relational tables and views as XML documents. The use of XML documents for insert, update, and delete operations to the database is also allowed. Oracle Database supports XML natively in the database, through Oracle XML DB, a distinct group of technologies related to high-performance XML storage and retrieval. Oracle XML DB is an evolution of the database that encompasses both SQL and XML data models in a highly interoperable manner, providing native XML support.

ODP.NET, Managed Driver follows XPath 1.0 specification and hence it does not support default XML namespaces. XML namespaces must be explicitly added to search or update nodes. This behavior differs from ODP.NET, Unmanaged Driver.

For samples related to ODP.NET XML support in ODAC installations done using Oracle Universal Installer, see the following directory:

`ORACLE_BASE\ORACLE_HOME\ODACsamples`

This section includes these topics:

- [Supported XML Features](#)
- [OracleXmlType and Connection Dependency](#)
- [Updating XMLType Data in the Database](#)
- [Updating XML Data in OracleXmlType](#)
- [Characters with Special Meaning in XML](#)
- [Retrieving Query Result Set as XML](#)
- [Data Manipulation Using XML](#)

Supported XML Features

XML support in ODP.NET provides the ability to do the following:

- Store XML data natively in the database as the Oracle database native type, `XMLType`.
- Access relational and object-relational data as XML data from an Oracle Database instance into the Microsoft .NET environment, and process the XML using the Microsoft .NET Framework.
- Save changes to the database using XML data.
- Execute XQuery statements.

For the .NET application developer, these features include the following:

- Enhancements to the `OracleCommand`, `OracleConnection`, and `OracleDataReader` classes.
- The following XML-specific classes:
 - `OracleXmlType`
`OracleXmlType` objects are used to retrieve Oracle native `XMLType` data.
 - `OracleXmlStream`
`OracleXmlStream` objects are used to retrieve XML data from `OracleXmlType` objects as a read-only .NET Stream object.
 - `OracleXmlQueryProperties`
`OracleXmlQueryProperties` objects represent the XML properties used by the `OracleCommand` class when the `XmlCommandType` property is `Query`.
 - `OracleXmlSaveProperties`
`OracleXmlSaveProperties` objects represent the XML properties used by the `OracleCommand` class when the `XmlCommandType` property is `Insert`, `Update`, or `Delete`.

See Also

- ["XQuery Support"](#)
- ["OracleCommand Class"](#)
- ["OracleXmlType Class"](#)
- ["OracleXmlStream Class"](#)
- ["OracleXmlQueryProperties Class"](#)
- ["OracleXmlSaveProperties Class"](#)
- *Oracle XML DB Developer's Guide*

XQuery Support

ODP.NET supports the XQuery language through a native implementation of SQL/XML functions, `XMLQuery` and `XMLTable`. When executing XQuery statements, Oracle XML DB generally evaluates XQuery expressions by compiling them into the same underlying structures as relational queries. Queries are optimized, leveraging both relational-database

and XQuery-specific optimization technologies, so that Oracle XML DB serves as a native XQuery engine. The treatment of all XQuery expressions, whether natively compiled or evaluated functionally, is transparent: programmers do not need to change their code to take advantage of XQuery optimizations.

① See Also

Oracle XML DB Developer's Guide to learn more about Oracle's XQuery support

OracleXmlType and Connection Dependency

The read-only `Connection` property of the `OracleXmlType` class holds a reference to the `OracleConnection` object used to instantiate the `OracleXmlType` class.

How the `OracleXmlType` object obtains a reference to an `OracleConnection` object depends on how the `OracleXmlType` class is instantiated:

- Instantiated from an `OracleDataReader` class using the `GetOracleXmlType`, `GetOracleValue`, or `GetOracleValues` method:
The `Connection` property is set with a reference to the same `OracleConnection` object used by the `OracleDataReader` object.
- Instantiated by invoking an `OracleXmlType` constructor with one of the parameters of type `OracleConnection`:
The `Connection` property is set with a reference to the same `OracleConnection` object provided in the constructor.
- Instantiated by invoking an `OracleXmlType(OracleClob)` constructor:
The `Connection` property is set with a reference to the `OracleConnection` object used by the `OracleClob` object.

An `OracleXmlType` object that is associated with one connection cannot be used with a different connection. For example, if an `OracleXmlType` object is obtained using `OracleConnection A`, that `OracleXmlType` object cannot be used as an input parameter of a command that uses `OracleConnection B`. By checking the `Connection` property of the `OracleXmlType` objects, the application can ensure that `OracleXmlType` objects are used only within the context of the `OracleConnection` referenced by its connection property. Otherwise, ODP.NET raises an exception.

Updating XMLType Data in the Database

Updating `XMLType` columns does not require a transaction. However, encapsulating the entire database update process within a transaction is highly recommended. This allows the updates to be rolled back if there are any errors.

`XMLType` columns in the database can be updated using Oracle Data Provider for .NET in a few ways:

- [Updating with DataSet, OracleDataAdapter, and OracleCommandBuilder](#)
- [Updating with OracleCommand and OracleParameter](#)

Updating with DataSet, OracleDataAdapter, and OracleCommandBuilder

If the XMLType column is fetched into the DataSet, the XMLType data is represented as a .NET String.

Modifying XMLType data in the DataSet does not require special treatment. XMLType data can be modified in the same way as any data that is stored in the DataSet. When a change is made and the OracleDataAdapter.Update method is invoked, the OracleDataAdapter object ensures that the XMLType data is handled properly. The OracleDataAdapter object uses any custom SQL INSERT, UPDATE, or DELETE statements that are provided. Otherwise, valid SQL statements are generated by the OracleCommandBuilder object as needed to [flush](#) the changes to the database.

Updating with OracleCommand and OracleParameter

The OracleCommand class provides a powerful way of updating XMLType data, especially with the use of an OracleParameter object. To update columns in a database table, the new value for the column can be passed as an input parameter of a command.

Input Binding

To update an XMLType column in the database, a SQL statement can be executed using static values. In addition, input parameters can be bound to SQL statements, anonymous PL/SQL blocks, or stored procedures to update XMLType columns. The parameter value can be set as .NET Framework Types, ODP.NET Types, or OracleXmlType objects.

While XMLType columns can be updated using an OracleXmlType object, having an instance of an OracleXmlType class does not guarantee that the XMLType column in the database can be updated.

Setting XMLType Column to NULL Value

Applications can set an XMLType column in the database to a NULL value, with or without input binding, as follows:

- Setting NULL values in an XMLType column with input binding

To set the XMLType column to NULL, the application can bind an input parameter whose value is DBNull.Value. This indicates to the OracleCommand object that a NULL value is to be inserted.

Passing in a null OracleXmlType object as an input parameter does not insert a NULL value into the XMLType column. In this case, the OracleCommand object raises an exception.

- Setting NULL Values in an XMLType Column without input binding

The following example demonstrates setting NULL values in an XMLType column without input binding:

```
// Create a table with an XMLType column in the database
CREATE TABLE XML_TABLE(NUM_COL number, XMLTYPE_COL xmltype);
```

An application can set a NULL value in the XMLType column by explicitly inserting a NULL value or by not inserting anything into that column as in the following examples:

```
insert into xml_table(xmltype_col) values(NULL);
```

```
update xml_table t set t.xmltype_col=NULL;
```

Setting XMLType Column to Empty XML Data

The XMLType column can be initialized with empty XML data, using a SQL statement:

```
// Create a table with an XMLType column in the database
CREATE TABLE XML_TABLE(NUM_COL number, XMLTYPE_COL xmltype);

INSERT INTO XML_TABLE (NUM_COL, XMLTYPE_COL) VALUES (4,
XMLType.createxml('<DOC/>'));
```

Updating XML Data in OracleXmlType

The following are ways that XML data can be updated in an OracleXmlType object.

- The XML data can be updated by passing an XPATH expression and the new value to the Update method on the OracleXmlType object.
- The XML data can be retrieved on the client side as the .NET Framework XmlDocument object using the GetXmlDocument method on the OracleXmlType object. This XML data can then be manipulated using suitable .NET Framework classes. A new OracleXmlType can be created with the updated XML data from the .NET Framework classes. This new OracleXmlType is bound as an input parameter to an update or insert statement.

Characters with Special Meaning in XML

The following characters in [Table 3-30](#) have special meaning in XML. For more information, refer to the XML 1.0 specifications

Table 3-30 Characters with Special Meaning in XML

Character	Meaning in XML	Entity Encoding
<	Begins an XML tag	<
>	Ends an XML tag	>
"	Quotation mark	"
'	Apostrophe or single quotation mark	'
&	Ampersand	&

When these characters appear as data in an XML element, they are replaced with their equivalent entity encoding.

Also certain characters are not valid in XML element names. When SQL identifiers (such as column names) are mapped to XML element names, these characters are converted to a sequence of hexadecimal digits, derived from the Unicode encoding of the character, bracketed by an introductory underscore, a lowercase x and a trailing underscore. A blank space is not a valid character in an XML element name. If a SQL identifier contains a space character, then in the corresponding XML element name, the space character is replaced by `_x0020_`, which is based on Unicode encoding of the space character.

Retrieving Query Result Set as XML

This section discusses retrieving the result set from a SQL query as XML data.

Handling Date and Time Format

The generated XML DATE and TIMESTAMP formats are based on the standard XML Schema formats.

See Also

<http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/datatypes.html#isoformats> for more information on the XML Schema specification.

Characters with Special Meaning in Column Data

If the data in any of the select list columns in the query contains any characters with special meaning in XML (see [Table 3-30](#)), these characters are replaced with their corresponding entity encoding in the result XML document.

The following examples demonstrate how ODP.NET handles the angle bracket characters in the column data:

```
/* Database Setup
connect scott/tiger@oracle
drop table specialchars;
create table specialchars ("id" number, name varchar2(255));
insert into specialchars values (1, '<Jones>');
commit;
*/

// C#

using System;
using System.Data;
using System.Xml;
using Oracle.DataAccess.Client;

class QueryResultAsXMLSample
{
    static void Main()
    {
        OracleConnection con = new OracleConnection();

        con.ConnectionString = "User Id=scott;Password=tiger;Data Source=oracle;";
        con.Open();

        // Create the command
        OracleCommand cmd = new OracleCommand("", con);

        // Set the XML command type to query.
        cmd.XmlCommandType = OracleXmlCommandType.Query;

        // Set the SQL query
        cmd.CommandText = "select * from specialchars";

        // Set command properties that affect XML query behavior.
        cmd.BindByName = true;

        // Set the XML query properties
        cmd.XmlQueryProperties.MaxRows = -1;
```

```

// Get the XML document as an XmlReader.
XmlReader xmlReader = cmd.ExecuteXmlReader();
XmlDocument xmlDocument = new XmlDocument();

xmlDocument.PreserveWhitespace = true;
xmlDocument.Load(xmlReader);
Console.WriteLine(xmlDocument.OuterXml);

// Close and Dispose OracleConnection object
con.Close();
con.Dispose();
}
}

```

The following XML document is generated for that table: The XML entity encoding that represents the angle brackets appears in bold.

```

<?xml version = '1.0'?>
<ROWSET>
  <ROW>
    <id>1</id>
    <NAME>&lt;Jones&gt;</NAME>
  </ROW>
</ROWSET>

```

Characters in Table or View Name

If a table or view name has any non-alphanumeric characters other than an underscore (_), the table or view name must be enclosed in quotation marks.

For example, to select all entries from a table with the name test'ing, the CommandText property of the OracleCommand object must be set to the following string:

```
"select * from \"test'ing\"";
```

Case-Sensitivity in Column Name to XML Element Name Mapping

The mapping of SQL identifiers (column names) to XML element names is case-sensitive, and the element names are in exactly the same case as the column names of the table or view.

However, the root tag and row tag names are case-insensitive. The following example demonstrates case-sensitivity in this situation:

```

//Create the following table
create table casesensitive_table ("Id" number, NAME varchar2(255));

//insert name and id
insert into casesensitive_table values(1, 'Smith');

```

The following XML document is generated:

```

<?xml version = '1.0'?>
<ROWSET>
  <ROW>
    <Id>1</Id>
    <NAME>Smith</NAME>
  </ROW>
</ROWSET>

```

Note that the element name for the Id column matches the case of the column name.

Column Name to XML Element Name Mapping

For each row generated by the SQL query, the SQL identifier (column name) maps to an XML element in the generated XML document, as shown in the following example:

```
// Create the following table
create table emp_table (EMPLOYEE_ID NUMBER(4), LAST_NAME varchar2(25));
// Insert some data
insert into emp_table values(205, 'Higgins');
```

The SQL query, `SELECT * FROM EMP_TABLE`, generates the following XML document:

```
<?XML version="1.0"?>
<ROWSET>
  <ROW>
    <EMPLOYEE_ID>205</EMPLOYEE_ID>
    <LAST_NAME>Higgins</LAST_NAME>
  </ROW>
</ROWSET>
```

The `EMPLOYEE_ID` and `LAST_NAME` database columns of the `employees` table map to the `EMPLOYEE_ID` and `LAST_NAME` elements of the generated XML document.

This section demonstrates how Oracle database handles the mapping of SQL identifiers to XML element names, when retrieving query results as XML from the database. The demonstration uses the `specialchars` table involving the `some id` column.

```
// Create the specialchars table
create table specialchars ("some id" number, name varchar2(255));
```

Note that the `specialchars` table has a column named `some id` that contains a blank space character. The space character is not allowed in an XML element name.

When retrieving the query results as XML, the SQL identifiers in the query select list can contain characters that are not valid in XML element names. When these SQL identifiers (such as column names) are mapped to XML element names, each of these characters is converted to a sequence of hexadecimal digits, derived from the Unicode encoding of the characters, bracketed by an introductory underscore, a lowercase x, and a trailing underscore.

Thus, the SQL query in the following example can be used to get a result as an XML document from the `specialchars` table:

```
select "some id", name from specialchars;
```

① See Also

["Characters with Special Meaning in XML"](#)

Improving Default Mapping

You can improve the default mapping of SQL identifiers to XML element names by using the following techniques:

- Modify the source. Create an object-relational view over the source schema, and make that view the new source.

- Use cursor subqueries and cast-multiset constructs in the SQL query.
- Create an alias for the column or attribute names in the SQL query. Prefix the aliases with an at sign (@) to map them to XML attributes instead of XML elements.
- Modify the XML document. Use [Extensible Stylesheet Language Transformation \(XSLT\)](#) to transform the XML document. Specify the XSL document and parameters. The transformation is done automatically after the XML document is generated from the relational data. Note that this may have an impact on performance.
- Specify the name of the root tag and row tag used in the XML document.

Object-Relational Data

ODP.NET can generate an XML document for data stored in object-relational columns, tables, and views, as shown in the following example:

```
// Create the following tables and types
CREATE TYPE "EmployeeType" AS OBJECT (EMPNO NUMBER, ENAME VARCHAR2(20));
/
CREATE TYPE EmployeeListType AS TABLE OF "EmployeeType";
/
CREATE TABLE mydept (DEPTNO NUMBER, DEPTNAME VARCHAR2(20),
                     EMPLIST EmployeeListType)
                     NESTED TABLE EMPLIST STORE AS EMPLIST_TABLE;
INSERT INTO mydept VALUES (1, 'depta',
                          EmployeeListType("EmployeeType"(1, 'empa')));
```

The following XML document is generated for the table:

```
<?xml version = "1.0"?>
<ROWSET>
  <ROW>
    <DEPTNO>1</DEPTNO>
    <DEPTNAME>depta</DEPTNAME>
    <EMPLIST>
      <EmployeeType>
        <EMPNO>1</EMPNO>
        <ENAME>empa</ENAME>
      </EmployeeType>
    </EMPLIST>
  </ROW>
</ROWSET>
```

ODP.NET encloses each item in a collection element, with the database type name of the element in the collection. The mydept table has a collection in the EMPLIST database column and each item in the collection is of type EmployeeType. Therefore, in the XML document, each item in the collection is enclosed in the type name EmployeeType, which appears in bold in the example.

NULL Values

If any database row has a column with a NULL value, then that column does not appear for that row in the generated XML document.

Data Manipulation Using XML

This section discusses making changes to the database data using XML.

Handling Date and Time Format

The generated XML DATE and TIMESTAMP formats are based on the standard XML Schema formats.

See Also

<http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/datatypes.html#isoformats> for more information on the XML Schema specification.

Saving Changes Using XML

Changes can be saved to database tables and views using XML data. However, insert, update, and delete operations cannot be combined in a single XML document. ODP.NET cannot accept a single XML document and determine which are insert, update, or delete changes.

The insert change must be in an XML document containing only rows to be inserted, the update changes only with rows to be updated, and the delete changes only with rows to be deleted.

For example, using the employees table that comes with the HR sample schema, you can specify the following query:

```
select employee_id, last_name from employees where employee_id = 205;
```

The following XML document is generated:

```
<?xml version = '1.0'?>
<ROWSET>
  <ROW>
    <EMPLOYEE_ID>205</EMPLOYEE_ID>
    <LAST_NAME>Higgins</LAST_NAME>
  </ROW>
</ROWSET>
```

To change the name of employee 205 from **Higgins** to **Smith**, specify the employees table and the XML data containing the changes as follows:

```
<?xml version = '1.0'?>
<ROWSET>
  <ROW>
    <EMPLOYEE_ID>205</EMPLOYEE_ID>
    <LAST_NAME>Smith</LAST_NAME>
  </ROW>
</ROWSET>
```

Characters with Special Meaning in Column Data

If the data in any of the elements in the XML document contains characters that have a special meaning in XML (see [Table 3-30](#)), these characters must be replaced with appropriate entity encoding, or be preceded by an escape character in the XML document, so that the data is stored correctly in the database table column. Otherwise, ODP.NET throws an exception.

The following example demonstrates how ODP.NET handles the angle bracket special characters in the column data, using entity encoding:

```
// Create the following table
create table specialchars ("id" number, name varchar2(255));
```

The following XML document can be used to insert values (1, '<Jones>') into the specialchars table. The XML entity encoding that represents the angle brackets appears in bold.

```
<?xml version = '1.0'?>
<ROWSET>
  <ROW>
    <id>1</id >
    <NAME>&lt;Jones&gt;</NAME>
  </ROW>
</ROWSET>
```

Characters with Special Meaning in Table or View Name

If a table or view name has any non-alphanumeric characters other than an underscore (_), the table or view name must be enclosed in quotation marks.

For example, to save changes to a table with the name test 'ing, the OracleCommand.XmlSaveProperties.TableName property must be set to "\"test 'ing\"".

Case-Sensitivity in XML Element Name to Column Name Mapping

For each XML element that represents a row of data in the XML document, the child XML elements map to database column names. The mapping of the child element name to the column name is always case-sensitive, but the root tag and row tag names are case-insensitive. The following example demonstrates this case-sensitivity:

```
//Create the following table
create table casesensitive_table ("Id" number, NAME varchar2(255));
```

The following XML document can be used to insert values (1, Smith) into the casesensitive_table:

```
<?xml version = '1.0'?>
<ROWSET>
  <ROW>
    <Id>1</Id>
    <NAME>Smith</NAME>
  </ROW>
</ROWSET>
```

Note that the element name for the Id column matches the case of the column name.

XML Element Name to Column Name Mapping

This section describes how Oracle database handles the mapping of XML element names to column names when using XML for data manipulation in the database. The following specialchars table involving the some id column demonstrates this handling.

```
// Create the specialchars table
create table specialchars ("some id" number, name varchar2(255));
```

Note that the specialchars table has a column named some id that contains a blank space character. The space character is not allowed in an XML element name.

Saving Changes to a Table Using an XML Document

When an XML document is used to save changes to a table or view, the `OracleCommand.XmlSaveProperties.UpdateColumnsList` property is used to specify the list of columns to update or insert.

When an XML document is used to save changes to a column in a table or view, and the corresponding column name contains any of the characters that are not valid in an XML element name, the escaped column name must be specified in the `UpdateColumnsList` property as in the following example.

The following XML document can be used to insert values (2, <Jones>) into the `specialchars` table:

```
<?xml version = '1.0'?>
<ROWSET>
  <ROW>
    <some_x0020_id>2</some_x0020_id>
    <NAME>&lt;Jones&gt;</NAME>
  </ROW>
</ROWSET>
```

The following example specifies the list of columns to update or insert:

```
/* Database Setup
connect scott/tiger@oracle
drop table specialchars;
create table specialchars ("some id" number, name varchar2(255));
insert into specialchars values (1, '<Jones>');
commit;
*/

// C#

using System;
using System.Data;
using System.Xml;
using Oracle.DataAccess.Client;

class InsertUsingXmlDocSample
{
    static void Main()
    {
        OracleConnection con = new OracleConnection();

        con.ConnectionString = "User Id=scott;Password=tiger;Data Source=oracle;";
        con.Open();
        Console.WriteLine("Connected Successfully");

        // Create the command
        OracleCommand cmd = new OracleCommand("", con);

        // Set the XML command type to query.
        cmd.XmlCommandType = OracleXmlCommandType.Insert;

        // Set the XML document
        cmd.CommandText = "<?xml version = '1.0'?>\n" + "<ROWSET>\n" + "<ROW>\n" +
            "<some_x0020_id>2</some_x0020_id>\n" + "<NAME>&lt;Jones&gt;</NAME>\n" +
            "</ROW>\n" + "</ROWSET>\n";
        cmd.XmlSaveProperties.Table = "specialchars";
```

```
string[] ucols = new string[2];

ucols[0] = "some_x0020_id";
ucols[1] = "NAME";
cmd.XmlSaveProperties.UpdateColumnsList = ucols;

// Insert rows
int rows = cmd.ExecuteNonQuery();

Console.WriteLine("Number of rows inserted successfully : {0} ", rows);

// Close and Dispose OracleConnection object
con.Close();
con.Dispose();
}
}
```

Improving Default Mapping

You can improve the default mapping by using the following techniques:

- Modify the target. Create an object-relational view over the target schema, and make the view the new target.
- Modify the XML document. Use XSLT to transform the XML document. Specify the XSL document and parameters. The transformation is done before the changes are saved. Note that this may have an impact on performance.
- Specify the name of the row tag used in the XML document.

Object-Relational Data

Changes in an XML document can also be saved to object-relational data. Each item in a collection can be specified in one of the following ways in the XML document:

- By enclosing the database type name of the item as the XML element name.
- By enclosing the name of the database column holding the collection with `_ITEM` appended as the XML element name.

Multiple Tables

Oracle Database does not save changes to multiple relational tables that have been joined together. Oracle recommends that you create a view on those relational tables, and then update that view. If the view cannot be updated, triggers can be used instead.

See Also

Oracle Database SQL Language Reference for the description and syntax of the `CREATE VIEW` statement

Commit Transactions

When the changes in an XML document are made, either all the changes are committed, or if an error occurs, all changes are rolled back.

Oracle User-Defined Types (UDTs) and .NET Custom Types

ODP.NET has the ability to represent Oracle UDTs found in the database as custom types in .NET applications. UDTs are useful in representing complex entities as a single object that can be shared among applications. Oracle products, such as Oracle Spatial and Oracle XML DB, use their own complex types frequently.

To represent Oracle UDTs as .NET custom types, applications must apply .NET attributes to custom classes and structs, and to their public fields and properties.

To convert between UDTs and custom types, ODP.NET uses custom interfaces.

Starting with version 21.2, managed ODP.NET and ODP.NET Core support Oracle UDTs and .NET custom types. These providers are near-parity with unmanaged ODP.NET UDT APIs and features, which makes migrating to managed or core more straightforward for developers. Only a few code changes may be needed.

This section discusses the ODP.NET UDT topics below. All the topics, except [Using UDTs with Managed ODP.NET and ODP.NET Core](#) and [Migrating from Unmanaged ODP.NET to Managed or Core](#), discuss unmanaged ODP.NET UDT features and Oracle UDTs generally. These two topics contrast unmanaged ODP.NET UDTs with managed ODP.NET and ODP.NET Core UDTs.

- [Oracle User-Defined Types \(UDTs\)](#)
- [Custom Types](#)
- [Specifying Custom Type Mappings](#)
- [Converting Between Custom Types and Oracle UDTs](#)
- [Oracle UDT Attribute Mappings](#)
- [Oracle UDT Retrieval from OracleDataReader](#)
- [Oracle UDT Metadata Retrieval from OracleDataReader](#)
- [Oracle UDT Parameter Binding with OracleParameter](#)
- [Populating the DataSet with Oracle UDTs](#)
- [UDT Method Invocation](#)
- [Configuration Settings for Oracle UDTs](#)
- [Using UDTs with Managed ODP.NET and ODP.NET Core](#)
- [Migrating from Unmanaged ODP.NET to Managed or Core](#)
- [Handling NULL Attribute Values in UDTs](#)

Oracle User-Defined Types (UDTs)

Oracle Data Provider for .NET supports Oracle object types or user-defined types (UDTs), which are defined in the Oracle database.

There are two kinds of UDTs:

- Object types (Oracle Object)
- Collection types (which can be VARRAY types or nested table types)

Additionally, ODP.NET supports references (REF) to object types.

The term UDT is used interchangeably with Oracle object types and abstract data types (ADTs).

The name of the Oracle UDT is case-sensitive and must be in the form *schema_name.type_name*.

① See Also

- [OracleRef Class](#)
- *Oracle AI Database Object-Relational Developer's Guide* for complete descriptions of object types
- UDT samples are available online on GitHub in the UDT directory:
<https://github.com/oracle/dotnet-db-samples/tree/master/samples>

Custom Types

Oracle Data Provider for .NET supports UDTs by representing Oracle UDTs defined in the database as .NET types, that is, custom types. For every Oracle UDT that the application wishes to fetch and manipulate, one custom type factory and one custom type are needed. The custom factory class is solely responsible for instantiating the custom type. ODP.NET uses the interfaces implemented on the custom factory classes to instantiate custom types at run time. Custom types define the mapping between the Oracle UDT attributes or elements to the .NET members. ODP.NET uses the interfaces implemented on the custom type instances to transfer values between the Oracle UDT and the custom type at run time.

Custom types can be .NET classes or structures. They can represent either Oracle Objects or Oracle Collections. Custom types can be implemented manually by the application developer or generated through an ODP.NET code generation tool.

Once the factory class and the custom type are defined and meet the implementation requirements, the application may set ODP.NET to automatically discover the mapping between the Oracle UDT and the custom type. This discovery process is based on the attribute that is applied on the custom factory class. Alternatively, the application can provide an explicit mapping through a configuration file.

Oracle Collections can be represented as an array of .NET Types. For example, an Oracle Collection type of NUMBER can be mapped to an `int[]`. Moreover, an Oracle Collection type of an Oracle UDT can be mapped to an array of the custom type.

Custom types must adhere to certain requirements in order for ODP.NET to represent Oracle UDTs as custom types. These requirements are as follows:

Required Custom Type Implementations

This section lists the required implementations for a custom .NET class or structure.

- `Oracle.DataAccess.Types.IOracleCustomType` interface implementation
This interface is used for conversions between custom types and Oracle UDTs.
The interface methods are implemented using the static methods of the `OracleUdt` class.
- Custom Type Factories

A custom type factory is used to create an instance of a custom type. A custom type factory is an implementation of either the `IOracleCustomTypeFactory` interface, the `IOracleArrayTypeFactory` interface, or both interfaces, as follows:

- To create a custom type that represents an Oracle Object, the custom type or a separate custom type factory class must implement the `Oracle.DataAccess.Types.IOracleCustomTypeFactory` interface.
 - To create a custom type that represents an Oracle Collection, the custom type or a separate custom type factory class must implement the `Oracle.DataAccess.Types.IOracleCustomTypeFactory` interface and the `Oracle.DataAccess.Types.IOracleArrayTypeFactory` interface.
 - To create an array type that represents an Oracle Collection, a custom type factory class must implement the `Oracle.DataAccess.Types.IOracleArrayTypeFactory` interface.
- Custom Type Member Mapping Attributes

The custom type member mapping attributes specify the mapping between custom type members and either Oracle object attributes or Oracle collection elements.

There are two types of custom type member mapping attributes:

- `OracleObjectMappingAttribute`

This attribute specifies the mapping between custom type members and Oracle object attributes for custom types that represent Oracle objects. This attribute must be applied to each custom type member (either field or property) that represents an Oracle Object attribute.

Note

Not all Oracle object attributes need to be mapped to custom type members. If there is no `OracleObjectMappingAttribute` for a particular object attribute, ODP.NET ignores that object attribute when converting between Oracle objects and custom types.

- `OracleArrayMappingAttribute`

This attribute specifies the custom type member that stores the elements of an Oracle collection for custom types representing Oracle collections. The attribute must be specified on only one of the custom type members.

- `Oracle.DataAccess.Types.INullable` interface implementation

This interface is used to determine if an instance of a custom type represents a null UDT. The `IsNull` property of the interface enables applications and ODP.NET to determine whether or not the UDT is null.

- Static Null field

The public static `Null` property is used to return a null UDT. This property returns a custom type with an `IsNull` property that returns true.

Optional Custom Type Implementations

The following are optional:

- `IXMLSerializable`

The `IXMLSerializable` interface is used in the .NET 2.0 framework to enable conversion between the custom type and its XML representation. This interface is only used if the serialization and deserialization of a custom type is needed in the `DataSet`.

- `Static Parse` and `Public ToString` methods

These methods enable conversion between the custom type and its string representation.

These methods are invoked when a `DataGrid` control is used to accept changes and display instance values.

- `Type Inheritance`

Type Inheritance refers to the process of deriving an Oracle UDT in the database from a super type.

If the custom type represents an Oracle UDT that is derived from a super type, the custom class should follow the same type hierarchy, that is, the custom class should be derived from another custom class that represents the super type defined in the database.

- `OracleCustomTypeMappingAttribute`

The `OracleCustomTypeMappingAttribute` object specifies the mapping between a custom type (or an array type) and an Oracle UDT.

There must be a unique custom type factory for each Oracle UDT used by the application as follows:

- `Oracle Object Types`:

The custom type factory must return a custom type that only represents the specified Oracle Object Type.

- `Oracle Collection Types`:

The custom type factory may return a custom type that can be used by other Oracle Collection Types. This is common when an array type is used to represent an Oracle Collection, for example, when an `int[]` is used to represent a collection of `NUMBERS`.

If the `OracleCustomTypeMappingAttribute` is not specified, then custom type mappings must be specified through XML configuration files, that is, `machine.config`, and either `app.config` for Windows applications or `web.config` for web applications.

See Also

- ["IOracleCustomType Interface"](#)
- ["OracleRef Class"](#)
- ["IOracleCustomTypeFactory Interface"](#)

Specifying Custom Type Mappings

After creating a custom type, the application must specify a custom type mapping that maps the custom type to an Oracle UDT in the database. This can be done using a custom type factory or XML in configuration files.

Using XML to specify custom type mappings has priority, if both techniques have been implemented. At run time, if ODP.NET finds custom type mappings specified in configuration files, it ignores any custom type mappings specified through the `OracleCustomTypeMappingAttribute` object. If a .NET application dynamically loads .NET

assemblies, which contain .NET classes that Oracle UDTs are mapped to, then the mapping between .NET classes and Oracle UDTs must be configured using a .NET config file.

Custom type mappings cannot be specified using synonyms, regardless of whether or not the mapping is provided through the `OracleCustomTypeMappingAttribute` object or the XML configuration file.

📘 See Also

Oracle Developer Tools for Visual Studio help sections on User-Defined Types Node under Server Explorer in Visual Studio for further information on UDT mapping.

This section contains these topics:

- ["Using a Custom Type Factory to Specify Custom Type Mappings"](#)
- ["Using XML in Configuration Files to Specify Custom Type Mappings"](#)

Using a Custom Type Factory to Specify Custom Type Mappings

The application can specify a custom type mapping using a custom type factory. The application supplies the name of the Oracle UDT, in the format `schema_name.type_name`, to an `OracleCustomTypeMappingAttribute` object and applies the name to the corresponding custom type factory. A custom type factory is a class or struct that implements either or both the `IOracleCustomTypeFactory` and `IOracleArrayTypeFactory` interfaces.

Note that for each Oracle UDT used by the application, there must be a unique custom type factory. Additionally, for Oracle Object Types, the custom type factory must return a custom type that uniquely represents the specified Oracle Object Type. For Oracle Collection Types, the custom type factory returns a custom type that can be used by other Oracle Collection Types. This is common when an custom type that is an array type represents an Oracle Collection, that is, when an `int[]` is used to represent a collection of NUMBERS.

At run time, using reflection programming, ODP.NET discovers all the custom type mappings specified by the application through the `OracleCustomTypeMappingAttribute` object.

📘 Note

The UDT name that is specified in the `OracleCustomTypeMappingAttribute` may not contain a period.

Using XML in Configuration Files to Specify Custom Type Mappings

The application can specify a custom type mapping with XML in configuration files, for example: using `machine.config`, and either `app.config` for Windows applications or `web.config` for web applications.

The custom type mappings must be specified in the `oracle.dataaccess.client` configuration section group. Each custom type mapping must be added to the collection of custom type mappings using the XML element `<add>`.

Each custom type mapping is consists of a name attribute and a value attribute. The name attribute may be any user-specified name that represents the custom type mapping. The value

attribute must begin with `udtMapping` and be followed by the required and optional attributes listed below.

Required Attributes

- `factoryName`

The case-sensitive assembly qualified name of the custom type factory class or struct.

If the assembly that defines the custom type factory does not have a strong name, then a partial assembly name consisting of just the assembly name is sufficient. In the case of strongly named assemblies, a complete assembly name is required. It must include the assembly name, the Version, Culture, `PublicKeyToken`.

- `typeName`

The case-sensitive name of the UDT defined in the database. By default all UDTs are created in the database with upper case names

- `schemaName`

The case-sensitive schema in which the UDT is defined in the database. By default all schemas are created in the database with upper case names

Optional Attributes

- `dataSource`

If specified, indicates that the custom type mapping applies only to Oracle UDTs defined in the database that the application connects to, as specified by the TNS name alias.

The Data Source is case-insensitive.

The following is an example of the format of the XML that can be specified in the configuration file for .NET 2.0:

```
<oracle.dataaccess.client>
  <settings>
    <add name="Person" value="udtMapping factoryName='Sample.PersonFactory,
      Sample, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null'
      typeName='PERSON' schemaName='SCOTT' dataSource='oracle'"/>
    <add name="Student" value="udtMapping factoryName='Sample.StudentFactory,
      Sample, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null'
      typeName='STUDENT' schemaName='SCOTT'"/>
  </settings>
</oracle.dataaccess.client>
```

Using Custom Type Mappings

During data retrieval, the application uses the custom type mappings to convert an Oracle UDT to a custom type. When data is provided back to the database through an input or input/output parameter, or by an update through an Oracle REF, the application uses the mappings to convert the custom type to an Oracle UDT.

In the case of input and input/output parameters, the application must also set the `OracleParameter.UdtTypeName` property to the user-defined type name of the parameter.

In certain cases, where Oracle UDTs are part of a type hierarchy, the custom type must be instantiated as a specific type in the type hierarchy. The Oracle UDT provided by the custom type mapping must be a subtype of the Oracle UDT specified by the `OracleParameter.UdtTypeName` property.

For example, the parameter for a stored procedure is of type, `SCOTT.PERSON` and has a subtype, `SCOTT.STUDENT`. The application has a custom class instance that represents `SCOTT.STUDENT`. The `UdtTypeName` is set to `SCOTT.PERSON`, but the custom type mapping indicates that the custom class is mapped to `SCOTT.STUDENT` and overrides the `UdtTypeName` when it instantiates the Oracle UDT. Thus, ODP.NET instantiates and binds Oracle UDTs appropriately when the custom object represents an Oracle UDT that is a subtype of the parameter type.

Converting Between Custom Types and Oracle UDTs

ODP.NET can convert between Oracle UDTs and custom types, if the proper attribute mappings are specified and the custom types are defined properly.

ODP.NET performs a conversion whenever an Oracle UDT is fetched as:

- In, out, in/out parameters bound for SQL or PL/SQL execution

The `DbType` property of `OracleParameter` must be set to `DbType.Object` or the `OracleDbType` property must be set to `OracleDbType.Object` or `OracleDbType.Array`.

For parameters that are user-defined types, the `UdtTypeName` property of the `OracleParameter` object must be always set to the parameter type.

Note: The `UdtTypeName` may differ from the Oracle UDT specified in the custom type mapping. This is the case when the parameter type is a super type of the Oracle UDT that the custom type represents.

- Column value retrieved from an `OracleDataReader` object

If the application requests for the value either through the `GetValue`, `GetValues`, `GetOracleValue`, `GetOracleValues`, `GetProviderSpecificValue`, or `GetProviderSpecificValues` methods or the `Item[]` property for a UDT column, ODP.NET finds the corresponding custom type that represents the Oracle UDT and carries out the proper conversion.

- Part of a `ResultSet` that populates the `DataSet`

If the application populates the `DataSet` with a result that contains UDTs using the `Fill` method on the `OracleDataAdapter`, the `DataSet` is populated with custom types that represent Oracle UDTs. With ADO.NET 2.0, the `DataSet` is populated with custom types for UDT columns regardless of whether the `ReturnProviderSpecificTypes` on the `OracleDataAdapter` is set to `true` or `false`.

- A `Object` referenced through a `REF`

When an `Object` referenced by a `REF` is retrieved, the custom type that represents the Oracle UDT is returned.

The application can use the `OracleUdtFetchOption` method to control the copy of the `Object` that is returned as follows:

- If the `OracleUdtFetchOption.Cache` option is specified and a cached copy of the object exists, the cached copy is immediately returned. If no cached copy exists, the latest object copy from the database is cached and returned.
- If the `OracleUdtFetchOption.Server` option is specified, the latest object copy from the database is cached and returned. If the object is already cached, the latest object copy overwrites the existing one.
- If the `OracleUdtFetchOption.TransactionCache` option is specified, there are two possibilities within the same transaction:

- * If the object copy was previously retrieved using the Server or TransactionCache option, the TransactionCache option behavior becomes equivalent to the Cache option behavior.
- * If the object copy was not previously retrieved using the Server or TransactionCache option, the TransactionCache option behavior becomes equivalent to the Server option behavior.

Oracle UDT Attribute Mappings

[Table 3-31](#) lists valid mappings of attributes (for objects) and elements (for collections), between Oracle UDT types and custom object types which can be either .NET types or Oracle provider-specific types (ODP.NET types).

Oracle collections do not have to map to a custom class. They can map to arrays of a specific type. [Table 3-31](#) indicates those collections with elements of a specified Oracle type that can map to arrays of a .NET Type or a provider-specific type. For example, if an Oracle Collection is a VARRAY of NUMBER(8), it can map to a `typeof(int[])`. This eliminates the need to construct a class that only holds an `int[]`.

For .NET 2.0, Oracle Collections can be mapped to Nullable types. This allows .NET 2.0 applications to obtain a nullable `int[]` which can hold null values in the `int[]`.

Note that Oracle UDT attributes and elements cannot be mapped to `object` or `object[]`.

Table 3-31 Attribute Mappings Between UDTs and Custom Object Types

Type of UDT Attribute or Element	.NET Type	ODP.NET Type
BFILE	System.Byte[]	OracleBFile
BINARY FLOAT	System.Byte, System.Int16, System.Int32, System.Int64, System.Single, System.Double, System.Decimal	OracleDecimal
BINARY DOUBLE	System.Byte, System.Int16, System.Int32, System.Int64, System.Single, System.Double, System.Decimal	OracleDecimal
BLOB	System.Byte[]	OracleBlob
CHAR	System.Char[], System.String	OracleString
CLOB	System.Char[], System.String	OracleClob
DATE	System.DateTime	OracleDate
INTERVAL DAY TO SECOND	System.TimeSpan,	OracleIntervalDS
INTERVAL YEAR TO MONTH	System.Int64	OracleIntervalYM
LONG RAW	System.Byte[]	OracleBinary

Table 3-31 (Cont.) Attribute Mappings Between UDTs and Custom Object Types

Type of UDT Attribute or Element	.NET Type	ODP.NET Type
NCHAR	System.Char[], System.String	OracleString
NCLOB	System.Char[], System.String	OracleClob
Nested Table	<i>custom type, .NET type[], or custom type[]</i>	<i>ODP Type[]</i>
NUMBER	System.Byte, System.Int16, System.Int32, System.Int64, System.Single, System.Double, System.Decimal	OracleDecimal
NVARCHAR2	System.Char[], System.String	OracleString
Object Type	<i>custom type</i>	N/A
RAW	System.Byte[]	OracleBinary
REF	System.String	OracleRef
TIMESTAMP	System.DateTime	OracleTimeStamp
TIMESTAMP WITH LOCAL TIME ZONE	System.DateTime	OracleTimeStampLTZ
TIMESTAMP WITH TIME ZONE	System.DateTime	OracleTimeStampTZ
VARCHAR2	System.Char[], System.String	OracleString
VARRAY	<i>custom type, .NET type[], or custom type[]</i>	<i>ODP Type[]</i>
XMLTYPE	System.Char[], System.String	OracleXmlType

Notes:

1. Conversion from a System.Byte[] to a BFILE is not supported, and therefore, System.Byte[] only represents a BFILE in read-only scenarios.

Oracle UDT Retrieval from OracleDataReader

In order to retrieve Oracle UDTs from the `OracleDataReader`, an application must specify a custom type mapping that determines the type that will represent the Oracle UDT. Once a custom type mapping has been specified and any necessary custom types have been created, the application can retrieve Oracle UDTs.

[Table 3-32](#) shows the type and value returned from an `OracleDataReader` object based on the method invoked, the column type, and whether or not there is a valid Custom type mapping.

Note

PS Object refers to a provider-specific object.

Table 3-32 Type and Value Returned from OracleDataReader Object

OracleDataReader method/ property invocation	Column Data Type	Custom Type Mapping	Value Returned for Oracle UDT	NULL Value Returned for Oracle UDT
Item[index], Item[name], GetValue(), GetValues()	Object, Collection	none	Exception thrown	Exception thrown
Item[index], Item[name], GetValue(), GetValues()	Object	<i>schema.type</i>	<i>custom object</i>	DBNull.Value
Item[index], Item[name], GetValue(), GetValues()	Collection	<i>schema.type</i>	<i>custom object</i> <i>custom object[]</i> <i>.NET Type[]</i> <i>PS</i> <i>object[]</i>	DBNull.Value
Item[index], Item[name], GetValue(), GetValues()	REF	none <i>schema.type</i>	<i>string</i> (HEX)	DBNull.Value
GetString()	REF	none <i>schema.type</i>	<i>string</i> (HEX)	Exception thrown
GetProviderSpecificValue(), GetProviderSpecificValues(), GetOracleValue(), GetOracleValues()	Object, Collection	<i>schema.type</i>	<i>custom object</i>	<i>custom type.Null</i>
GetProviderSpecificValue(), GetProviderSpecificValues(), GetOracleValue(), GetOracleValues()	Collection	<i>schema.type</i>	<i>custom object[]</i> <i>.NET Type[]</i> <i>PS</i> <i>object[]</i>	null
GetProviderSpecificValue(), GetProviderSpecificValues(), GetOracleValue(), GetOracleValues(), GetOracleRef()	REF	none <i>schema.type</i>	OracleRef	OracleRef.Null
GetOracleString()	REF	none <i>schema.type</i>	OracleString (HEX)	OracleString.Null

See Also

["Obtaining Data from an OracleDataReader Object"](#)

Oracle UDT Metadata Retrieval from OracleDataReader

An `OracleDataReader` object can return metadata used to determine the custom type that represents an Oracle UDT when a .NET Type or Provider-Specific Type accessor is invoked.

The same custom type is used when populating the DataSet using the `OracleDataAdapter.Fill` method.

[Table 3-33](#) shows the values returned from the `OracleDataReader.GetFieldType` and `GetProviderSpecificFieldType` methods that specify the .NET type of the column.

Table 3-33 Values Returned from OracleDataReader Methods

OracleDataReader Method/ Property invocation	Column Data Type	Custom Type Mapping	Return Value
<code>GetFieldType(index)</code>	Object, Collection	none	Exception thrown
<code>GetFieldType(index)</code>	Object	<i>schema.type</i>	<code>typeof(custom type)</code>
<code>GetFieldType(index)</code>	Collection	<i>schema.type</i>	<code>typeof(custom type) typeof(custom type[]) typeof(.NET type[]) typeof(PS type[])</code>
<code>GetFieldType(index)</code>	REF	none <i>schema.type</i>	<code>typeof(string)</code>
<code>GetProviderSpecificField Type(index)</code>	Object, Collection	none	Exception thrown
<code>GetProviderSpecificField Type(index)</code>	Object,	<i>schema.type</i>	<code>typeof(custom type)</code>
<code>GetProviderSpecificField Type(index)</code>	Collection	<i>schema.type</i>	<code>typeof(custom type) typeof(custom type[]) typeof(.NET type[]) typeof(PS type[])</code>
<code>GetProviderSpecificField Type(index)</code>	REF	none <i>schema.type</i>	<code>typeof(OracleRef)</code>

Oracle UDT Parameter Binding with OracleParameter

This section discusses using UDT output and input parameter bindings with an `OracleParameter` object.

① See Also

["Parameter Binding"](#)

This section contains these topics:

- [Guidelines for Binding UDT Input and Output Parameters](#)
- [UDT Input Parameter Binding with OracleParameters](#)
- [UDT Output Parameter Binding with OracleParameters](#)

Guidelines for Binding UDT Input and Output Parameters

Developers must consider the following when using UDT parameter bindings with an `OracleParameter` object.

- The `UdtTypeName` property must be set. Binding is based on the `UdtTypeName` property regardless of the parameter direction.

Note

The `UdtTypeName` may differ from the Oracle UDT specified in the custom type mapping. This occurs when the parameter type is a super type of the Oracle UDT that the custom type represents.

- In case of Input/Output binding, the behavior is the same as Input and Output parameters.
- For Input parameter values, the bind value is converted to the UDT specified by the custom type mapping.
- For Output parameters:
 - If the value being returned is an Oracle Object or Collection, it is converted to a custom type or array type as specified by the custom type mapping. The value returned is always a custom type or an array type, regardless of whether the property most recently set was `DbType` or `OracleDbType`.
 - If the value being returned is a REF, then no custom type mapping is required.

UDT Input Parameter Binding with OracleParameters

Only certain combinations of these `OracleParameter` property values, `DbType`, `OracleDbType`, and `UdtTypeName`, can exist on the `OracleParameter` object. `OracleParameter` objects cannot be set to combinations that are not listed.

[Table 3-34](#) describes the valid ways of binding input parameters for Oracle UDTs.

The last column indicates the Oracle type that ODP.NET converts the `OracleParameter` value to before binding.

Table 3-34 Valid Ways to Bind Input Parameters for Oracle UDTs

OracleParameter. Value	OracleParameter. DbType or OracleParameter. OracleDbType	OracleParameter. UdtTypeName	Custom Type Mappings	Oracle Type converted to before Binding
<i>custom object</i> <i>custom object[]</i> <i>.NET object[]</i> <i>PS object[]</i> String (HEX) OracleString(HEX) OracleRef	DbType.Object OracleDbType.Object OracleDbType.Array OracleDbType.Ref	not set	none <i>schema.type</i>	Exception thrown
<i>custom object[]</i> <i>.NET object[]</i> <i>PS object[]</i>	DbType.Object OracleDbType.Object OracleDbType.Array	<i>schema.type</i>	none	Exception thrown
<i>custom object</i>	DbType.Object	<i>schema.type</i>	<i>schema.type</i>	Specified UDT is instantiated. Value is bound as Object or Collection, based on the <code>UdtTypeName</code> property

Table 3-34 (Cont.) Valid Ways to Bind Input Parameters for Oracle UDTs

OracleParameter. Value	OracleParameter. DbType or OracleParameter. OracleDbType	OracleParameter. UdtTypeName	Custom Type Mappings	Oracle Type converted to before Binding
<i>custom object</i>	OracleDbType.Object	<i>schema.type</i>	<i>schema.type</i>	Specified UDT is instantiated. <i>schema.type</i> must represent an object.
<i>custom object</i>	OracleDbType.Array	<i>schema.type</i>	<i>schema.type</i>	Specified UDT is instantiated. <i>schema.type</i> must represent a collection.
.NET object[] PS object[] custom object[]	DbType.Object OracleDbType.Array	<i>schema.type</i>	<i>schema.type</i>	UDT specified by OracleParameter.UdtTypeName is instantiated.
.NET object[] PS object[] custom object[]	OracleDbType.Object	<i>schema.type</i>	none <i>schema.type</i>	Exception thrown
<i>custom object</i> .NET object[] PS object[] custom object[]	OracleDbType.Ref	<i>schema.type</i>	none <i>schema.type</i>	Exception thrown
String (HEX) OracleString (HEX) OracleRef	DbType.Object OracleDbType.Object OracleDbType.Array	<i>schema.type</i>	none <i>schema.type</i>	Exception thrown
Char[] (HEX) String (HEX) OracleString (HEX) OracleRef	OracleDbType.Ref	<i>schema.type</i>	none <i>schema.type</i>	A REF

UDT Output Parameter Binding with OracleParameters

Only certain combinations of these OracleParameter property values, DbType, OracleDbType, and UdtTypeName, can exist on the OracleParameter object. OracleParameter objects cannot be set to combinations that are not listed.

[Table 3-35](#) shows the supported ODP.NET output parameter bindings of Oracle database objects.

The last column indicates the type that ODP.NET converts the OracleParameter value to before binding.

Table 3-35 Valid Ways to Bind Output Parameters for Oracle UDTs

Type returned from Oracle	OracleParameter. DbType	OracleParameter. UdtTypeName	Custom Type Mappings	Type converted to
Object/ Collection/REF	DbType.Object OracleDbType.Object OracleDbType.Array OracleDbType.Ref	not set	none <i>schema.type</i>	Exception thrown

Table 3-35 (Cont.) Valid Ways to Bind Output Parameters for Oracle UDTs

Type returned from Oracle	OracleParameter.DbType	OracleParameter.UdtTypeName	Custom Type Mappings	Type converted to
Object/Collection	DbType.Object OracleDbType.Object OracleDbType.Array	<i>schema.type</i>	none	Exception thrown
Object	DbType.Object OracleDbType.Object	<i>schema.type</i>	<i>schema.type</i>	<i>custom object</i>
Object	OracleDbType.Array OracleDbType.Ref	<i>schema.type</i>	none <i>schema.type</i>	Exception thrown
Collection	OracleDbType.Array DbType.Object	<i>schema.type</i>	<i>schema.type</i>	<i>custom object</i> <i>custom object[]</i> <i>.NET object[]</i> <i>PS object[]</i>
Collection	OracleDbType.Ref OracleDbType.Object	<i>schema.type</i>	none <i>schema.type</i>	Exception thrown
REF	DbType.Object OracleDbType.Object OracleDbType.Array	<i>schema.type</i>	none <i>schema.type</i>	Exception thrown
REF	OracleDbType.Ref	<i>schema.type</i>	none <i>schema.type</i>	OracleRef

① See Also

- ["Parameter Binding"](#)
- ["Typed OracleDataReader Accessors"](#)

Populating the DataSet with Oracle UDTs

The DataSet is a disconnected result set. With ADO.NET 2.0, both .NET types and provider-specific types can be used to populate the DataSet. This section describes the types used to populate the DataSet when the column is an Oracle UDT.

[Table 3-36](#) lists the types that populate the DataSet column, based on the Oracle column type, the ReturnProviderSpecificTypes property of the DataAdapter, the existence of a custom type mapping, the DataSet column type, the DataSet column value, and the DataSet column null value.

Table 3-36 Types that Populate the DataSet with ADO.NET 2.0

Oracle Column Type	ReturnProviderSpecificTypes Property	Custom Type Mappings	DataSet Column Type	DataSet Column Value	DataSet Column Null Value
Object / Collection	False/True	none	Exception thrown	Exception thrown	Exception thrown
Object / Collection	False	<i>schema.type</i>	<i>typeof(custom type)</i>	<i>custom object</i>	DBNull.Value

Table 3-36 (Cont.) Types that Populate the DataSet with ADO.NET 2.0

Oracle Column Type	ReturnProvider-SpecificTypes Property	Custom Type Mappings	DataSet Column Type	DataSet Column Value	DataSet Column Null Value
Object / Collection	True	schema.type	typeof(custom type)	custom object	custom object.Null
Collection	False	schema.type	typeof(custom type[]) typeof(.NET type[]) typeof(PS type[])	.NET type[] PS object[] custom object[]	DBNull.Value
Collection	True	schema.type	typeof(custom type[]) typeof(.NET type[]) typeof(PS type[])	.NET type[] PS object[] custom object[]	null
REF	False	none schema.type	typeof(string)	string/HEX	DBNull.Value
REF	True	none schema.type	typeof(OracleRef)	OracleRef	OracleRef.Null

UDT Method Invocation

ODP.NET supports invocation of methods defined for a UDT on the database. This can be accomplished by doing the following:

1. Set the CommandType as CommandType.StoredProcedure.
2. Set the CommandText as "type_name.procedure_name"
3. Execute the command using any of the Execute methods on the OracleCommand object.

For instance functions, the parameters are as follows:

- The first parameter must be the return value.
- The second parameter must be the UDT instance on which the instance method is invoked, which is the instance of the .NET custom object.
- Subsequent parameters are for the function.

For instance procedures, the first parameter must be the UDT instance.

For static methods, the UDT instance is not needed.

Configuration Settings for Oracle UDTs

ODP.NET exposes two configuration settings to determine how ODP.NET handles Oracle UDTs.

- [StatementCacheWithUdts](#)
- [UdtCacheSize](#)

These configuration settings can be specified as machine-wide settings for a particular version of ODP.NET, using the registry key with the name that exists under HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\ODP.NET\Assembly_Version. The configuration settings specified in the registry can be overridden if an entry is created in the machine.config

for .NET framework-wide settings, or in the `app.config` or `web.config` for application-specific settings.

See Also

[Configuring Oracle Data Provider for .NET](#) for details on configuring ODP.NET.

StatementCacheWithUdts

`StatementCacheWithUdts` specifies whether or not ODP.NET caches Oracle UDTs retrieved by a `SELECT` statement along with the statement when it is returned to the statement cache. Possible values are 1 - Yes (the default) or 0 - No.

For the value of 1, the Oracle UDTs are cached along with the statements. Therefore, the memory that contained the UDTs can be re-used; subsequent executions of the same statement do not require additional memory. This may result in an overall higher performance.

For the value of 0, ODP.NET frees the memory for the retrieved Oracle UDTs before the statement is returned to the statement cache. This may result in poorer performance because subsequent executions will require new memory allocations.

UdtCacheSize

`UdtCacheSize` specifies the size of the object cache for each connection that ODP.NET uses when retrieving and manipulating Oracle UDTs. The value for this setting must be specified in kilobytes (KB) with the default 4096KB, equivalent to 4 MB.

This configuration setting is used to determine how frequently the objects in the object cache will be purged (using an LRU approach) as the limit of the object cache size approaches.

Using UDTs with Managed ODP.NET and ODP.NET Core

Managed ODP.NET and ODP.NET Core support the same or similar UDT APIs and features as unmanaged ODP.NET. The differences are listed in this documentation section.

Managed ODP.NET and ODP.NET Core UDT features require connecting to Oracle Database 12.1 or higher versions.

XML Configuration Files Not Supported

Managed ODP.NET and ODP.NET Core do not support Oracle custom type mapping via XML configuration files, such as in `app.config` or `web.config`. For these providers, custom type mapping occurs through using the `OracleCustomTypeMapping` attribute.

OracleUdt FromCustomObject and ToCustomObject Method Changes

The unmanaged ODP.NET `OracleUdt FromCustomObject` and `ToCustomObject` method declarations are as follows:

- `FromCustomObject(OracleConnection con, IntPtr pObject)`
- `ToCustomObject(OracleConnection con, IntPtr pObject)`

For managed ODP.NET and ODP.NET Core, the declarations are, respectively:

- `FromCustomObject(OracleConnection con, object pObject)`

- `ToCustomObject(OracleConnection con, object pObject)`

The second parameter is an object instead of a pointer.

OracleUdt Static Methods

`OracleUdt` static methods, `GetValue`, `IsDBNull`, and `SetValue`, and their overloads all have one `InPtr` parameter. For managed ODP.NET and ODP.NET Core, the `InPtr` is replaced with an object that represents the UDT. In unmanaged ODP.NET, the static methods are as follows:

- `GetValue`
 - `GetValue(OracleConnection con, IntPtr pUdt, string attrName)`
 - `GetValue(OracleConnection con, IntPtr pUdt, int attrIndex)`
 - `GetValue(OracleConnection con, IntPtr pUdt, string attrName, out object statusArray)`
 - `GetValue(OracleConnection con, IntPtr pUdt, int attrIndex, out object statusArray)`
- `IsDBNull`
 - `IsDBNull(OracleConnection con, IntPtr pUdt, string attrName)`
 - `IsDBNull(OracleConnection con, IntPtr pUdt, int attrIndex)`
- `SetValue`
 - `SetValue(OracleConnection con, IntPtr pUdt, string attrName, object value)`
 - `SetValue(OracleConnection con, IntPtr pUdt, int attrIndex, object value)`
 - `SetValue(OracleConnection con, IntPtr pUdt, string attrName, object value, object statusArray)`
 - `SetValue(OracleConnection con, IntPtr pUdt, int attrIndex, object value, object statusArray)`

The managed ODP.NET and ODP.NET Core static methods are, respectively:

- `GetValue`
 - `GetValue(OracleConnection con, object udt, string attrName)`
 - `GetValue(OracleConnection con, object udt, int attrIndex)`
 - `GetValue(OracleConnection con, object udt, string attrName, out object statusArray)`
 - `GetValue(OracleConnection con, object udt, int attrIndex, out object statusArray)`
- `IsDBNull`
 - `IsDBNull(OracleConnection con, object udt, string attrName)`
 - `IsDBNull(OracleConnection con, object udt, int attrIndex)`
- `SetValue`
 - `SetValue(OracleConnection con, object udt, string attrName, object value)`
 - `SetValue(OracleConnection con, object udt, int attrIndex, object value)`
 - `SetValue(OracleConnection con, object udt, string attrName, object value, object statusArray)`

- SetValue(OracleConnection con, object udt, int attrIndex, object value, object statusArray)

OracleConnection Class Changes

Managed ODP.NET and ODP.NET Core do not support the following OracleConnection method:

- FlushCache()

OracleRef Class Changes

Managed ODP.NET and ODP.NET Core do not support object caches and other functionality without a managed code implementation. Thus, the following unmanaged ODP.NET class members are *not* supported in managed and core:

- Constructors
 - OracleRef(OracleConnection conn, string udtTypeName, string objTableName)
 - OracleRef(OracleConnection conn, string hexStr)
- Methods
 - Flush()
 - GetCustomObject(OracleUdtFetchOption fetchOption, int depthLevel)
 - GetCustomObjectForUpdate(bool bWait)
 - GetCustomObjectForUpdate(bool bWait, int depthLevel)
 - Lock(bool bWait)
- Properties
 - HasChanges {get}
 - ObjectTableName {get}

The following OracleRef class members are supported in a modified manner:

Table 3-37 OracleRef class members

Method in Unmanaged	Equivalent Core/Managed Method	Core/Managed Behavior
Delete(bool bFlush)	Delete()	Equivalent to Delete(true) in unmanaged ODP.NET.
GetCustomObject(OracleUdtFetchOption fetchOption)	GetCustomObject()	Always retrieves object from database. No object cache.
Update(object customObject, bool bFlush)	Update(object customObject)	Equivalent to Update(customObject, true) in unmanaged ODP.NET.

Migrating from Unmanaged ODP.NET to Managed or Core

The following changes are required to migrate UDT applications from unmanaged ODP.NET to either managed ODP.NET or ODP.NET Core:

- Assembly: Reference managed ODP.NET or ODP.NET Core assembly (i.e. `Oracle.ManagedDataAccess.dll`) in your .NET project.
- Namespace: Use the `Oracle.ManagedDataAccess.*` namespace in your source files.
- Configuration: Use `OracleCustomTypeMapping` attribute to define the custom type mapping.
- API: Use the managed/core `OracleUdt FromCustomObject` and `ToCustomObject` methods in the custom class.
- API: Use the managed/core `OracleUdt` static methods.
- API: Change `OracleRef` class invocation, including constructors, properties, and methods, to the members managed/core `OracleRef` class use.

Handling NULL Attribute Values in UDTs

A UDT attribute can be mapped to either a .NET type or to a provider-specific type.

To be able to properly represent NULL UDT attribute values as .NET types within the .NET custom classes, they can be defined as nullable .NET types. For example, if the UDT has a DATE attribute that can be NULL, then it can be mapped to a `DateTime? nullable` type rather than `DateTime`. This is due to fact that `DateTime` cannot be used to represent NULL values.

But when it comes to mapping UDT attributes as provider-specific types, then the .NET custom classes should simply have the non-nullable provider-specific type, for example, `OracleDate`. Each provider type can represent a NULL value through the provider-specific type's `Null` static field, for example, `OracleDate.Null`.

Bulk Copy

ODP.NET provides a Bulk Copy feature which enables applications to efficiently load large amounts of data from a table in one database to another table in the same or a different database.

The ODP.NET Bulk Copy feature uses a direct path load approach, which is similar to, but not the same as Oracle SQL*Loader. Using direct path load is faster than conventional loading (using conventional SQL INSERT statements). Direct path loading formats Oracle data blocks and writes the data blocks directly to the data files. Bulk Copy eliminates considerable processing overhead.

The ODP.NET Bulk Copy feature can load data into older Oracle databases.

The ODP.NET Bulk Copy feature is subject to the same basic restrictions and integrity constraints for direct path loads, as discussed in the next few sections.

ODP.NET Bulk Copy supports local transactions.

When using DRCP implicit connection pooling with a transaction boundary (`POOL_BOUNDARY=TRANSACTION`), explicitly start a local transaction before invoking `OracleBulkCopy.WriteToServer()`. ODP.NET's default behavior is auto-commit. Internal bulk copy SQL execution will auto-commit without a transaction and then return its session to the pool, which can cause errors for LOBs and cursors that depend on the session context.

① See Also

[System Requirements](#) to learn which versions of the Oracle AI Database ODP.NET interoperates with

Data Types Supported by Bulk Copy

Bulk Copy supports the following Oracle database data types:

- NUMBER
- BINARY_DOUBLE
- BINARY_FLOAT
- CHAR
- JSON
- NCHAR
- VARCHAR2
- NVARCHAR2
- LONG
- CLOB
- BLOB
- DATE
- TIMESTAMP
- TIMESTAMP WITH TIME ZONE
- TIMESTAMP WITH LOCAL TIME ZONE
- INTERVAL YEAR TO MONTH
- INTERVAL DAY TO SECOND
- BOOLEAN
- VECTOR

① Note

BOOLEAN and VECTOR are supported for managed ODP.NET and ODP.NET Core only

Bulk copy does not support overwrites.

ODP.NET converts the .NET source data into an Oracle format based on the destination table column definitions. If the source data is too large to be stored in the destination column, then the data may be truncated or converted to allow it to be stored in the database. For example, if the .NET string source is "abc" and the destination column is VARCHAR2(2), the data is truncated to "ab" when saved to the column.

Restrictions on Oracle Bulk Copy of a Single Partition

- The table that contains the partition cannot have any global indexes defined on it.
- The tables that the partition is a member of cannot have referential and check constraints enabled.
- Enabled triggers are not allowed.

Integrity Constraints Affecting Oracle Bulk Copy

During a Oracle bulk copy, some integrity constraints are automatically enabled or disabled, as follows:

Enabled Constraints

During an Oracle bulk copy, the following constraints are automatically enabled by default:

- NOT NULL
- UNIQUE
- PRIMARY KEY (unique-constraints on not-null columns)
- CHECK constraints

NOT NULL constraints are checked at column array build time. Any row that violates the NOT NULL constraint is rejected.

UNIQUE constraints are verified when indexes are rebuilt at the end of the load. The index is left in an Index Unusable state if it violates a UNIQUE constraint.

To enforce primary key and unique constraints, use the `EnforceIndexChecks` member in `OracleBulkCopyOptions`. Bulk copy will not load any rows and leave indexes unchanged when it detects an index-related error.

Disabled Constraints

During an Oracle bulk copy, the following constraints are automatically disabled by default:

- Referential constraints (FOREIGN KEY)

Database Insert Triggers

Table insert triggers are disabled when a direct path load begins. After the rows are loaded and indexes rebuilt, any triggers that were disabled are automatically reenabled. The log file lists all triggers that were disabled for the load. There should be no errors reenabling triggers.

Unlike integrity constraints, insert triggers are not reapplied to the whole table when they are enabled. As a result, insert triggers do not fire for any rows loaded on the direct path. When using the direct path, the application must ensure that any behavior associated with insert triggers is carried out for the new rows.

Field Defaults

Default column specifications defined in the database are not available with direct path loading. Fields for which default values are desired must be specified with the `DEFAULTIF` clause. If a

DEFAULTIF clause is not specified and the field is NULL, then a null value is inserted into the database.

① **See Also**

[Oracle Data Provider for .NET Bulk Copy Classes](#)

Asynchronous Programming and Pipelining

Asynchronous programming and Oracle Database pipelining remove performance bottlenecks and enhance overall responsiveness that can limit synchronous apps.

Starting with Release 26ai, managed ODP.NET and ODP.NET Core, support both asynchronous programming and database pipelining. Both these technologies are simple for developers to learn and incorporate into their existing or new ODP.NET apps.

Asynchronous Programming

Synchronous provider database operations, such as opening a connection, executing a query, or reading data, may take significant time to complete. Single-threaded applications block other operations to wait for the original operation to finish. In contrast, assigning the long-running operation to a background thread allows the foreground thread to remain active throughout. The long-running operation no longer blocks other app operations thereby realizing overall improved performance from using asynchronous behavior.

Asynchronous operations are ideal for long running operations whose results or completion are not needed immediately. Use it when user interfaces must remain as responsive as possible or when other operations can execute simultaneously without relying on the asynchronous operation completing.

.NET has a language-level asynchronous programming model. It allows writing asynchronous code without juggling callbacks nor conforming to an asynchronous library, which have historically complicated asynchronous .NET code. This model is called Task-based Asynchronous Pattern (TAP) and developers have found it easiest to develop with compared to earlier asynchronous patterns.

TAP uses the `async` modifier to specify an asynchronous method. When calling an `async` method, a task is returned. When the `await` operator is applied to the task, the current method exits immediately. When the task finishes, execution resumes in the same method. Most importantly, calling an `async` method does not allocate additional threads. Asynchronous functionality can be selectively added to existing ODP.NET apps to iteratively provide better runtime user experience.

Managed ODP.NET and ODP.NET Core support TAP starting with version 23. The feature is backwards compatible with Oracle Database 19c and higher. ODP.NET `async` APIs are identical to standard ADO.NET `async` APIs. Only `OracleBulkCopy.WriteToServerAsync` method is not currently supported. This makes developing ODP.NET `async` apps easy for experienced ADO.NET developers.

Oracle extends `async` capabilities to ODP.NET-specific classes as well, including `OracleBlob`, `OracleClob`, `OracleBFile`, and `OracleXmlStream`. They have `async` methods for reading, copying, and writing data as these data types can have very large sizes with their operations I/O bound.

Asynchronous ODP.NET sample code:

```
static async void Main(){
    OracleConnection oc = new OracleConnection(connectionString);

    // Establish a connection, asynchronously
    Task task = oc.OpenAsync(CancellationToken.None);

    // Execute operation(s) that do not require the connection
    Console.WriteLine("Hello World");
    OracleCommand cmd = oc.CreateCommand();
    cmd.CommandText = " select * from employees";

    // "await" OpenAsync completion before executing operations needing connection
    await task;

    // Execute the command
    OracleDataReader = await cmd.ExecuteReaderAsync();
}
```

ODP.NET Async Methods

OracleConnection:

```
public Task OpenAsync()

public override Task OpenAsync(CancellationToken cancellationToken)

public Task OpenWithNewPasswordAsync(string newPassword)

public Task OpenWithNewPasswordAsync(string newPassword, CancellationToken
cancellationToken)

public Task OpenWithNewPasswordAsync(SecureString secureNewPassword)

public Task OpenWithNewPasswordAsync(SecureString secureNewPassword,
CancellationToken cancellationToken)
```

OracleCommand:

```
public Task<int> ExecuteNonQueryAsync()

public override Task<int> ExecuteNonQueryAsync(CancellationToken
cancellationToken)

public Task<OracleDataReader> ExecuteReaderAsync()

public Task<OracleDataReader> ExecuteReaderAsync(CancellationToken
cancellationToken);

public Task<OracleDataReader> ExecuteReaderAsync(CommandBehavior behavior)

public Task<OracleDataReader> ExecuteReaderAsync(CommandBehavior behavior,
CancellationToken cancellationToken)

public Task<object> ExecuteScalarAsync()

public override Task<object> ExecuteScalarAsync(CancellationToken
cancellationToken)

public Task<XmlReader> ExecuteXmlReaderAsync()

public Task<XmlReader> ExecuteXmlReaderAsync(CancellationToken cancellationToken)
```

OracleDataReader:

```
public Task<bool> IsDBNullAsync(int i)

public override Task<bool> IsDBNullAsync(int i, CancellationToken
cancellationToken)

public Task<T> GetFieldValueAsync<T>(int i)

public override Task<T> GetFieldValueAsync<T>(int i, CancellationToken
cancellationToken)

public Task<bool> NextResultAsync()

public override Task<bool> NextResultAsync(CancellationToken cancellationToken)

public Task<bool> ReadAsync()

public override Task<bool> ReadAsync(CancellationToken cancellationToken)
```

OracleDataSource:

```
public ValueTask<DbConnection> OpenConnectionAsync(CancellationToken
cancellationToken)

public void DisposeAsync()
```

OracleBlob:

```
public Task<Int64> CopyToAsync(OracleBlob obj)

public Task<Int64> CopyToAsync(OracleBlob obj, CancellationToken
cancellationToken)

public Task<Int64> CopyToAsync(OracleBlob obj, Int64 dst_offset)

public Task<Int64> CopyToAsync(OracleBlob obj, Int64 dst_offset,
CancellationToken cancellationToken)

public Task<Int64> CopyToAsync(Int64 src_offset, OracleBlob obj, Int64
dst_offset, Int64 amount)

public Task<Int64> CopyToAsync(Int64 src_offset, OracleBlob obj, Int64
dst_offset, Int64 amount, CancellationToken cancellationToken)

public Task<int> ReadAsync(byte[] buffer, int offset, int count)

public override Task<int> ReadAsync(byte[] buffer, int offset, int count,
CancellationToken cancellationToken)

public Task<int> WriteAsync(byte[] buffer, int offset, int count)

public override Task<int> WriteAsync(byte[] buffer, int offset, int count,
CancellationToken cancellationToken)
```

OracleClob:

```
public Task<Int64> CopyToAsync(OracleClob obj)

public Task<Int64> CopyToAsync(OracleClob obj, CancellationToken
cancellationToken)

public Task<Int64> CopyToAsync(OracleClob obj, Int64 dst_offset)

public Task<Int64> CopyToAsync(OracleClob obj, Int64 dst_offset,
CancellationToken cancellationToken)
```

```
public Task<Int64> CopyToAsync(Int64 src_offset, OracleClob obj, Int64
dst_offset, Int64 amount)

public Task<Int64> CopyToAsync(Int64 src_offset, OracleClob obj, Int64
dst_offset, Int64 amount, CancellationToken cancellationToken)

public Task<int> ReadAsync(byte[] buffer, int offset, int count)

public override Task<int> ReadAsync(byte[] buffer, int offset, int count,
CancellationToken cancellationToken)

public Task<int> ReadAsync(char[] buffer, int offset, int count)

public Task<int> ReadAsync(char[] buffer, int offset, int count,
CancellationToken cancellationToken)

public Task<int> WriteAsync(byte[] buffer, int offset, int count)

public override Task<int> WriteAsync(byte[] buffer, int offset, int count,
CancellationToken cancellationToken)

public Task<int> WriteAsync(char[] buffer, int offset, int count)

public Task<int> WriteAsync(char[] buffer, int offset, int count,
CancellationToken cancellationToken)
```

OracleBFile:

```
public Task<Int64> CopyToAsync(OracleBlob obj)

public Task<Int64> CopyToAsync(OracleBlob obj, CancellationToken
cancellationToken)

public Task<Int64> CopyToAsync(OracleBlob obj, Int64 dst_offset)

public Task<Int64> CopyToAsync(OracleBlob obj, Int64 dst_offset,
CancellationToken cancellationToken)

public Task<Int64> CopyToAsync(Int64 src_offset, OracleBlob obj, Int64
dst_offset, Int64 amount)

public Task<Int64> CopyToAsync(Int64 src_offset, OracleBlob obj, Int64
dst_offset, Int64 amount, CancellationToken cancellationToken)

public Task<Int64> CopyToAsync(OracleClob obj)

public Task<Int64> CopyToAsync(OracleClob obj, CancellationToken
cancellationToken)

public Task<Int64> CopyToAsync(OracleClob obj, Int64 dst_offset)

public Task<Int64> CopyToAsync(OracleClob obj, Int64 dst_offset,
CancellationToken cancellationToken)

public Task<Int64> CopyToAsync(Int64 src_offset, OracleClob obj, Int64
dst_offset, Int64 amount)

public Task<Int64> CopyToAsync(Int64 src_offset, OracleClob obj, Int64
dst_offset, Int64 amount, CancellationToken cancellationToken)

public Task<int> ReadAsync(byte[] buffer, int offset, int count)
```

```
public override Task<int> ReadAsync(byte[] buffer, int offset, int count,
CancellationTokentoken cancellationToken)
```

OracleXmlStream:

```
public Task<int> ReadAsync(byte[] buffer, int offset, int count)

public override Task<int> ReadAsync(byte[] buffer, int offset, int count,
CancellationTokentoken cancellationToken)

public Task<int> ReadAsync(char[] buffer, int offset, int count)

public Task<int> ReadAsync(char[] buffer, int offset, int count,
CancellationTokentoken cancellationToken)
```

Database Pipelining

Asynchronous ODP.NET calls send requests to the database and read the responses on the client side asynchronously. On the database server, operations are independent of the client and are synchronous by default.

Database pipelining, a new Oracle AI Database 26ai feature, provides asynchronous execution capability on the database server side. Subsequent operations can send requests even before previous operations responses are received. ODP.NET will still receive results in the same order as their respective commands were submitted to the database.

The key runtime benefit is higher throughput and performance. The database is ready to receive new ODP.NET commands as they arrive instead of blocking until after the previous command completes.

Pipelining requires using ODP.NET asynchronous APIs. Apps can add pipelining to async ODP.NET apps with virtually no code changes. Functionally, there is no difference for the client application when pipelining is enabled or disabled since it is a database server capability.

Managed ODP.NET and ODP.NET Core support database pipelining. By default, the feature is disabled. To enable it, set the OracleConfiguration Pipelining property to true. Alternatively for managed ODP.NET, change the .NET configuration file Pipelining setting to true. No database setup is required other than using Oracle AI Database 26ai or higher.

ODP.NET Pipelining sample code:

```
static async void Main(){

    //Enable Pipelining
    OracleConfiguration.Pipelining = true;

    OracleConnection oc = new OracleConnection(connectionString);
    await oc.OpenAsync(CancellationTokentoken.None);

    OracleCommand cmd = oc.CreateCommand();
    OracleCommand cmd2 = oc.CreateCommand();

    cmd.CommandText = "update table1 set col1 = 1 where col2 = 2";
    cmd2.CommandText = "update table2 set col3 = 3 where col4 = 4";

    //Execute commands asynchronously with pipelining
    Task<int> task = cmd.ExecuteNonQueryAsync(CancellationTokentoken.None);
    Task<int> task2 = cmd2.ExecuteNonQueryAsync(CancellationTokentoken.None);

    //Execute other operations that do not require query results
    Console.WriteLine("Hello World");
}
```

```
//Await the asynchronous tasks to complete
int updatedRows = await task;
int updatedRows2 = await task2;

Console.WriteLine("Number updated rows are " + updatedRows + " and " + updatedRows2);
}
```

Async ODP.NET command cancellation by the `OracleCommand Cancel` method call, command timeout and `CancellationToken` is not supported when pipelining is enabled. When pipelining is disabled, async ODP.NET command cancellation and timeout are supported and has the same behavior as synchronous ODP.NET command cancellation and timeout.

Advanced Queuing and Transactional Event Queues

Oracle Database Advanced Queuing (AQ) and Oracle Transactional Event Queues (TxEventQ) provide database-integrated message queuing functionality. AQ leverages the functions of Oracle Database so that messages can be stored persistently, propagated between queues on different computers and databases, and transmitted using Oracle Net Services and HTTP(S). TxEventQ provides robust real-time messaging, streaming events, and publish/subscribe messaging with multiple publishers and multiple consumers. It is a high performance partitioned implementation with multiple event streams per queue, while AQ is a disk-based implementation for simpler workflow use cases.

ODP.NET uses the same APIs when executing operations against either AQ or TxEventQ. From a .NET application level, both queue types have the same functionality. Thus, ODP.NET developers can use the same code for working with either AQ or TxEventQ. Where the two queue types differ is in their implementations and administration.

Due to the commonality at the .NET app level, this book will use the term "AQ" to refer synonymously to both AQ and TxEventQ. When differences exist, the book will refer to each queue type separately.

ODP.NET queuing APIs execute operational functions, such as enqueueing and dequeuing messages. Administrative functions, such as creating and managing queues, are not available in ODP.NET. Use Oracle Developer Tools for Visual Studio or DBMS_AQADM PL/SQL package to perform these administrative functions.

As Oracle Database AQ is implemented in database tables, all operational benefits of high availability, scalability, and reliability are also applicable to queue data. Oracle Database AQ supports standard database features such as recovery, restart, and security.

The following items discuss Oracle Database AQ concepts:

- Queues and Queue Tables

Messages enqueued in a queue are stored in a queue table. A queue table must be created before creating a queue based on it. Use the DBMS_AQADM PL/SQL package or Oracle Developer Tools for Visual Studio to create and administer queue tables and queues.

Queues are represented by `OracleAQQueue` objects.

- Single-Consumer and Multiple-Consumer Queues

A single-consumer queue is created based on a single consumer queue table. Messages enqueued in a single-consumer queue can be dequeued by only a single consumer.

A multiple-consumer queue is based on a multiple-consumer queue table. This queue supports queue subscribers and message recipients.

- Message Recipients

A message producer can submit a list of recipients when enqueueing a message. This allows for a unique set of recipients for each message in the queue. The recipient list associated with the message overrides the subscriber list, if any, associated with the queue. The recipients need not be in the subscriber list. However, recipients can be selected from among the subscribers. The `Recipients` property of an `OracleAQMessage` can be used to specify the recipients to a specific message in terms of `OracleAQAgent` objects.

- Enqueue

Messages are enqueued when producer applications push the messages into a queue. This is accomplished by calling the `Enqueue` method on an `OracleAQQueue` object. Multiple messages can be enqueued using the `EnqueueArray` method.

- Dequeue

Messages are dequeued when consumer applications pull the messages from a queue. This is accomplished by calling the `Dequeue` method on an `OracleAQQueue` object. Multiple messages can be dequeued using the `DequeueArray` method.

- Listen

Subscriber applications can use a `Listen` call to monitor multiple queues for subscriptions on different queues. This is a more scalable solution for cases where a subscriber application has subscribed to many queues and wishes to receive messages that arrive in any of the queues. This is accomplished by calling the `Listen` method of the `OracleAQQueue` class, passing the list of subscriptions in form of an array.

- Notification

Subscriber applications can utilize the notification mechanism to get notifications about message availability in a queue. The applications can decide to skip or dequeue the message from the queue based on the information received.

A subscriber application must register for event notification on the queues from which it wants to receive notifications. This is represented by the `MessageAvailable` event on `OracleAQQueue`. The event is triggered when messages matching the subscriptions arrive.

Notifications can be registered as regular or grouping notifications. A time out value for these notifications can also be specified. Various notification options can be set using the `OracleAQQueue.Notification` property. Notifications set on an `OracleAQQueue` object gets cancelled automatically when the object gets disposed.

- Buffered Messaging

In buffered messaging, messages reside in a shared memory area. This makes it faster than persistent messaging. The messages are written to disk only when the total memory consumption of buffered messages approaches the available shared memory limit. Buffered messaging is ideal for applications that do not require the reliability and transaction support of Oracle Database AQ persistent messaging.

Buffered and persistent messages use the same single-consumer or multi-consumer queues, and the same administrative and operational interfaces. They are distinguished from each other by a delivery mode parameter. When an application enqueues a message to an Oracle Database AQ queue, it sets the delivery mode parameter as well.

- Messages

Messages consist of control information (metadata) and a payload (data). The control information represents message properties AQ uses to manage messages. The payload data is the information stored in the queue and is transparent to Oracle AQ. A message can reside in only one queue. A message is created by the `enqueue` call and consumed by the `dequeue` call.

The possible message payload data types are:

- JSON
- RAW
- User-Defined Type
- XML

Unmanaged ODP.NET does not support JSON payloads.

See Also

- ["OracleAQQueue Class"](#)
- ["Recipients"](#)
- ["OracleAQAgent Class"](#)
- ["Enqueue"](#)
- ["EnqueueArray"](#)
- ["Dequeue"](#)
- ["DequeueArray"](#)
- ["Listen"](#)
- ["MessageAvailable Event"](#)
- ["Notification"](#)
- ["DeliveryMode"](#)
- *Oracle AI Database Transactional Event Queues and Advanced Queuing User's Guide*

Using ODP.NET for Advanced Queuing

.NET applications can use ODP.NET to access all the operational features of AQ such as Enqueuing, Dequeuing, Listen, and Notification.

[Table 3-38](#) maps the AQ features to their corresponding ODP.NET implementation.

Table 3-38 Mapping AQ Features with their ODP.NET Implementation

Functionality	ODP.NET Implementation
Create a Message	Create an <code>OracleAQMessage</code> object
Enqueue a single message	Specify the message as <code>OracleAQMessage</code> , queue as <code>OracleAQQueue</code> and enqueue options on <code>OracleAQQueue</code> , call <code>OracleAQQueue.Enqueue</code>
Enqueue multiple messages	Specify the messages as an <code>OracleAQMessage</code> array in <code>OracleAQQueue.EnqueueArray</code>
Dequeue a single message	Specify dequeue options on <code>OracleAQQueue</code> and call <code>OracleAQQueue.Dequeue</code>
Dequeue multiple messages	Call <code>OracleAQQueue.DequeueArray</code>
Listen for messages on Queue(s)	Call <code>OracleAQQueue.Listen</code> . To listen on multiple queues use static <code>Listen</code> method of <code>OracleAQQueue</code>

Table 3-38 (Cont.) Mapping AQ Features with their ODP.NET Implementation

Functionality	ODP.NET Implementation
Message Notifications	Use <code>OracleAQQueue.MessageAvailable Event</code> along with the <code>NotificationConsumers</code> property

Enqueuing and Dequeuing Example

The following example demonstrates enqueueing and dequeuing messages using a single consumer queue. The first part of the example performs the requisite database setup for the database user, SCOTT. The second part of the example demonstrates enqueueing and dequeuing messages.

-- Part I: Database setup required for this demo

```

-----
-- SQL to grant appropriate privilege to database user, SCOTT
-----
SQL> ALTER USER SCOTT ACCOUNT UNLOCK IDENTIFIED BY Pwd4Sct;
User altered.
SQL> GRANT ALL ON DBMS_AQADM TO scott;

-----
-- PL/SQL to create queue-table and queue and start queue for SCOTT
-----
BEGIN
  DBMS_AQADM.CREATE_QUEUE_TABLE(
    queue_table=>'scott.test_q_tab',
    queue_payload_type=>'RAW',
    multiple_consumers=>FALSE);

  DBMS_AQADM.CREATE_QUEUE(
    queue_name=>'scott.test_q',
    queue_table=>'scott.test_q_tab');

  DBMS_AQADM.START_QUEUE(queue_name=>'scott.test_q');
END;
/

-----
-- PL/SQL to stop queue and drop queue & queue-table from SCOTT
-----
BEGIN
  DBMS_AQADM.STOP_QUEUE('scott.test_q');

  DBMS_AQADM.DROP_QUEUE(
    queue_name => 'scott.test_q',
    auto_commit => TRUE);

  DBMS_AQADM.DROP_QUEUE_TABLE(
    queue_table => 'scott.test_q_tab',
    force => FALSE,
    auto_commit => TRUE);
END;
/
-- End of Part I, database setup.

//Part II: Enqueueing and dequeuing messages

```

```
//C#
using System;
using System.Text;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

namespace ODPSample
{
    /// <summary>
    /// Demonstrates Enqueuing and Dequeuing raw message
    /// using a single consumer queue
    /// </summary>
    class EnqueueDequeue
    {
        static void Main(string[] args)
        {
            // Create connection
            string constr = "user id=scott;password=Pwd4Sct;data source=oracle";
            OracleConnection con = new OracleConnection(constr);

            // Create queue
            OracleAQQueue queue = new OracleAQQueue("scott.test_q", con);

            try
            {
                // Open connection
                con.Open();

                // Begin txn for enqueue
                OracleTransaction txn = con.BeginTransaction();

                // Set message type for the queue
                queue.MessageType = OracleAQMessageType.Raw;

                // Prepare message and RAW payload
                OracleAQMessage enqMsg = new OracleAQMessage();
                byte[] bytePayload = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
                enqMsg.Payload = bytePayload;

                // Prepare to Enqueue
                queue.EnqueueOptions.Visibility = OracleAQVisibilityMode.OnCommit;

                // Enqueue message
                queue.Enqueue(enqMsg);

                Console.WriteLine("Enqueued Message Payload      : "
                    + ByteArrayToString(enqMsg.Payload as byte[]));
                Console.WriteLine("MessageId of Enqueued Message : "
                    + ByteArrayToString(enqMsg.MessageId));

                // Enqueue txn commit
                txn.Commit();

                // Begin txn for Dequeue
                txn = con.BeginTransaction();

                // Prepare to Dequeue
                queue.DequeueOptions.Visibility = OracleAQVisibilityMode.OnCommit;
                queue.DequeueOptions.Wait = 10;

                // Dequeue message
                OracleAQMessage deqMsg = queue.Dequeue();
```

```

        Console.WriteLine("Dequeued Message Payload      : "
            + ByteArrayToString(deqMsg.Payload as byte[]));
        Console.WriteLine("MessageId of Dequeued Message : "
            + ByteArrayToString(deqMsg.MessageId));

        // Dequeue txn commit
        txn.Commit();
    }
    catch (Exception e)
    {
        Console.WriteLine("Error: {0}", e.Message);
    }
    finally
    {
        // Close/Dispose objects
        queue.Dispose();
        con.Close();
        con.Dispose();
    }
}

// Function to convert byte[] to string
static private string ByteArrayToString(byte[] byteArray)
{
    StringBuilder sb = new StringBuilder();
    for (int n = 0; n < byteArray.Length; n++)
    {
        sb.Append((int.Parse(byteArray[n].ToString())).ToString("X"));
    }
    return sb.ToString();
}
}
}

```

Continuous Query Notification Support

Oracle Data Provider for .NET provides a notification framework that supports Continuous Query Notification, enabling applications to receive client-side notifications when there is a change in a query result set, schema objects, or the state of the database, even if no Oracle Data Provider for .NET database connection exists. Using Continuous Query Notification, an application can maintain the validity of the client-side cache (for example, the ADO.NET DataSet) easily. Continuous Query Notification was previously known as Database Change Notification.

Using the notification framework, applications can specify a query result set as a registered query for notification request on the database, and create this notification registration to maintain the validity of the query result set. When there is a change on the database that could affect the client-side cache's query results, the notification framework notifies the application.

Note

The content of a change notification is referred to as an *invalidation message*. It indicates that the query result set is now invalid and provides information about the changes.

Based on the information provided by the invalidation message, the application can then act accordingly. For example, the application might need to refresh its own copy of the data for the registered query that is stored locally in the application.

Note

If a registered object is dropped from the database and a new one is created with the same name in the same schema, re-registration is required to receive notifications for the newly created object.

Firewalls, such as Windows Firewall, may be set up to block TCP network ports, which blocks incoming database notifications. Ensure the firewall is configured so that database applications can use the designated port for Continuous Query Notification. If the firewall configuration cannot be changed to allow notifications be sent to the client, consider enabling Client Initiated Continuous Query Notifications.

Continuous Query Notification queries can be query-based (default) or object-based. The query-based registrations allow ODP.NET to notify applications when the selected rows have changed in the database. The object-based registrations allow ODP.NET to notify applications for any changes that occur in the table(s) containing the selected rows.

Query-based registrations have two modes: guaranteed mode and best-effort mode. In guaranteed mode, any continuous query notification ensures that a change occurred to something contained in the queried result set. However, if a query is complex, then it cannot be registered in guaranteed mode. Best-effort mode is used in such cases.

Best-effort mode simplifies the query for query-based registration. No notifications are lost from the simplification. However, the simplification may cause false positives, as the simpler version's query result could change when the original query result would not. There still remain some restrictions on which queries can have best-effort mode query-based registrations. In such cases, developers can use object-based registrations, which can register most query types. Object-based registrations generate notifications when the query object changes, even if the actual query result does not. This also means that object-based registrations are more prone to false positives than query-based registrations. Developers should be aware of the relative strengths and weaknesses of each continuous query notification option and choose the one that best suits their requirements.

If a large number of rows are modified at once, consuming significant shared pool resources, the application will not receive any change notifications with specific row information that had undergone changes. Rather, it will receive a notification with `OracleNotificationEventArgs.Info` property set to `OracleNotificationInfo.Error`.

This section contains the following topics:

- [Client Initiated Continuous Query Notifications](#)
- [Continuous Query Notification Classes](#)
- [Supported Operations](#)
- [Requirements of Notification Registration](#)
- [Using Continuous Query Notification](#)
- [Continuous Query Notification Support](#)
- [Best Practice Guidelines and Performance Considerations](#)

① See Also

- [Configuring a Port to Listen for Database Notifications](#)
- *Oracle AI Database Administrator's Reference for Microsoft Windows* for details on configuring the Windows Firewall
- *Oracle AI Database Development Guide* for more information on Continuous Query Notification

Client Initiated Continuous Query Notifications

ODP.NET 21c introduces support for Client Initiated Continuous Query Notifications (CICQN). Traditional Continuous Query Notification (CQN) communicates database change notifications back to clients using a listening end point, a client's IP/hostname and a port. In cloud deployments or when firewalls between the database server and client cannot be configured to permit messages to the listening end point, these notifications will be blocked.

CICQN uses a dedicated connection instead of the listening end point. This connection aggregates all the database change notifications for the pool's users. ODP.NET attempts to always keep this connection open. It does not count toward Min Pool Size and Max Pool Size limits.

ODP.NET 23ai supports CICQN with Oracle Database 19c and higher. ODP.NET 21c CICQN requires Oracle Database 21c or higher. It can be enabled by setting the `OracleConfiguration.UseClientInitiatedCQN` static Boolean property or .NET Framework configuration file `UseClientInitiatedCQN` setting to true. By default, it is false.

For most on-premises deployments, traditional CQN is recommended.

Continuous Query Notification Classes

The following classes are associated with Continuous Query Notification Support:

- `OracleDependency`

Represents a dependency between an application and an Oracle database based on the database events which the application is interested in. It contains information about the dependency and provides the mechanism to notify the application when specified database events occurs. The `OracleDependency` class is also responsible for creating the notification listener to listen for database notifications. There is only one database notification listener for each application domain. This notification listener terminates when the application process terminates.

The dependency between the application and the database is not established when the `OracleDependency` object is created. The dependency is established when the command that is associated with this `OracleDependency` object is executed. That command execution creates a continuous query notification registration in the database.

When a change has occurred in the database, the `HasChanges` property of the `OracleDependency` object is set to true. Furthermore, if an event handler was registered with the `OnChange` event of the `OracleDependency` object, the registered event handler function will be invoked.

- `OracleNotificationRequest`

Represents a notification request to be registered in the database. It contains information about the request and the properties of the notification.

- `OracleNotificationEventArgs`

Represents the [invalidation message](#) generated for a notification when a specified database event occurs and contains details about that database event.

See Also

- ["OracleDependency Class"](#)
- ["OracleNotificationRequest Class"](#)
- ["OracleNotificationEventArgs Class"](#)

Supported Operations

The ODP.NET notification framework in conjunction with Continuous Query Notification supports the following activities:

- Creating a notification registration by:
 - Creating an `OracleDependency` instance and binding it to an `OracleCommand` instance.
- Grouping multiple notification requests into one registration by:
 - Using the `OracleDependency.AddCommandDependency` method.
 - Setting the `OracleCommand.Notification` request using the same `OracleNotificationRequest` instance.
- Registering for Continuous Query Notification by:
 - Executing the `OracleCommand`. If either the notification property is null or `NotificationAutoEnlist` is false, the notification will not be made.
- Removing notification registration by:
 - Using the `OracleDependency.RemoveRegistration` method.
 - Setting the `Timeout` property in the `OracleNotificationRequest` instance before the registration is created.
 - Setting the `IsNotifiedOnce` property to true in the `OracleNotificationRequest` instance before the registration is created. The registration is removed once a database notification is sent.
- Ensuring Change Notification Persistence by:
 - Specifying whether or not the invalidation message is queued persistently in the database before delivery. If an invalidation message is to be stored persistently in the database, then the change notification is guaranteed to be sent. If an invalidation message is stored in an in-memory queue, the change notification can be received faster, however, it could be lost upon database shutdown or crashes.
- Retrieving notification information including:
 - The changed object name.
 - The schema name of the changed object.
 - Database events that cause the notification, such as insert, delete, and so on.

- The RowID of the modified object row.

In Oracle SQL, the `ROWIDTOCHAR(ROWID)` and `ROWIDTONCHAR(ROWID)` functions convert a ROWID value to VARCHAR2 and NVARCHAR data types, respectively. If these functions are used within a SQL statement, ROWIDs are not returned in the `OracleNotificationEventArgs` object that is passed to the continuous query notification callback.

- Defining the listener port number.

By default, the static `OracleDependency.Port` property is set to -1. This indicates that the ODP.NET listens on a port that is randomly picked when ODP.NET registers a continuous query notification request for the first time during the execution of an application.

ODP.NET creates only one listener that listens on one port within an application domain. Once ODP.NET starts the listener, the port number cannot be changed; Changes to the static `OracleDependency.Port` property will generate an error if a listener has already been created.

See Also

- ["OracleCommand Class"](#)
- ["Notification"](#)
- ["NotificationAutoEnlist"](#)
- ["OracleDependency Class"](#)
- ["OracleNotificationEventArgs Class"](#)

Requirements of Notification Registration

The connected user must have the `CHANGE NOTIFICATION` privilege to create a notification registration.

This SQL statement grants the `CHANGE NOTIFICATION` privilege:

```
grant change notification to user name
```

This SQL statement revokes the `CHANGE NOTIFICATION` privilege:

```
revoke change notification from user name
```

Using Continuous Query Notification

This section describes what the application should do, and the flow of the process, when an application uses Continuous Query Notification to receive notifications for any changes in the registered query result set.

Application Steps

The application should do the following:

1. Create an `OracleDependency` instance.
2. Assign an event handler to the `OracleDependency.OnChange` event property if the application wishes to have an event handler invoked when database changes are

detected. Otherwise, the application can choose to poll on the HasChanges property of the OracleDependency object. This event handler is invoked when the change notification is received.

3. Set the port number for the listener to listen on. The application can specify the port number for one notification listener to listen on. If the application does not specify a port number, a random one is used by the listener.
4. Bind the OracleDependency instance to an OracleCommand instance that contains the actual query to be executed. Internally, the Continuous Query Notification request (an OracleNotificationRequest instance) is created and assigned to the OracleCommand.Notification property.

Flow of Notification Process

1. When the command associated with the notification request is executed, the notification registration is created in the database. The command execution must return a result set, or contain one or more REF cursors for a PL/SQL stored procedure.
2. ODP.NET starts the application listener on the first successful notification registration.
3. When a change related to the registration occurs in the database, the application is notified through the event delegate assigned to the OracleDependency.OnChange event property, or the application can poll the OracleDependency.HasChanges property.

The following example demonstrates the continuous query notification feature.

```
// Database Setup
// NOTE: unless the following SQL command is executed,
// ORA-29972 will be obtained from running this sample
/*
grant change notification to scott;
*/
using System;
using System.Threading;
using System.Data;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

//This sample shows the continuous query notification feature in ODP.NET.
//Application specifies to get a notification when emp table is updated.
//When emp table is updated, the application will get a notification
//through an event handler.
namespace NotificationSample
{
    public class MyNotificationSample
    {
        public static bool IsNotified = false;

        public static void Main(string[] args)
        {
            //To Run this sample, make sure that the change notification privilege
            //is granted to scott.
            string constr = "User Id=scott;Password=tiger;Data Source=oracle";
            OracleConnection con = null;
            OracleDependency dep = null;

            try
            {
                con = new OracleConnection(constr);
                OracleCommand cmd = new OracleCommand("select * from emp", con);
                con.Open();
```

```

// Set the port number for the listener to listen for the notification
// request
OracleDependency.Port = 1005;

// Create an OracleDependency instance and bind it to an OracleCommand
// instance.
// When an OracleDependency instance is bound to an OracleCommand
// instance, an OracleNotificationRequest is created and is set in the
// OracleCommand's Notification property. This indicates subsequent
// execution of command will register the notification.
// By default, the notification request is using the Database Change
// Notification.
dep = new OracleDependency(cmd);

// Add the event handler to handle the notification. The
// OnMyNotification method will be invoked when a notification message
// is received from the database
dep.OnChange +=
    new OnChangeEventHandler(MyNotificationSample.OnMyNotificaton);

// The notification registration is created and the query result sets
// associated with the command can be invalidated when there is a
// change. When the first notification registration occurs, the
// notification listener is started and the listener port number
// will be 1005.
cmd.ExecuteNonQuery();

// Updating emp table so that a notification can be received when
// the emp table is updated.
// Start a transaction to update emp table
OracleTransaction txn = con.BeginTransaction();
// Create a new command which will update emp table
string updateCmdText =
    "update emp set sal = sal + 10 where empno = 7782";
OracleCommand updateCmd = new OracleCommand(updateCmdText, con);
// Update the emp table
updateCmd.ExecuteNonQuery();
//When the transaction is committed, a notification will be sent from
//the database
txn.Commit();
}
catch (Exception e)
{
    Console.WriteLine(e.Message);
}

con.Close();
// Loop while waiting for notification
while(MyNotificationSample.IsNotified == false)
{
    Thread.Sleep(100);
}
}

public static void OnMyNotificaton(object src,
    OracleNotificationEventArgs arg)
{
    Console.WriteLine("Notification Received");
    DataTable changeDetails = arg.Details;
    Console.WriteLine("Data has changed in {0}",
        changeDetails.Rows[0]["ResourceName"]);
}

```

```
        MyNotificationSample.IsNotified = true;
    }
}
```

Best Practice Guidelines and Performance Considerations

This section provides guidelines for working with Continuous Query Notification and the ODP.NET notification framework, and discusses the performance impacts. Every change notification registration consumes database memory, storage or network resources, or some combination thereof. The resource consumption further depends on the volume and size of the [invalidation message](#). In order to scale well with a large number of mid-tier clients, Oracle recommends that the client implement these best practices:

- Few and mostly read-only tables
There should be few registered objects, and these should be mostly read-only, with very infrequent invalidations. If an object is extremely volatile, then a large number of invalidation notifications are sent, potentially requiring a lot of space (in memory or on disk) in the invalidation queue. This is also true if a large number of objects are registered.
- Few rows updated for each table
Transactions should update (or insert or delete) only a small number of rows within the registered tables. Depending on database resources, a whole table could be invalidated if too many rows are updated within a single transaction, for a given table.

This policy helps to contain the size of a single invalidation message, and reduces disk storage for the invalidation queue.

OracleDataAdapter Safe Type Mapping

The ODP.NET `OracleDataAdapter` class provides the Safe Type Mapping feature to ensure that the following Oracle data types do not lose data when converted to their closely related .NET types in the `DataSet`:

- NUMBER
- DATE
- TimeStamp (refers to all TimeStamp objects)
- INTERVAL DAY TO SECOND

Note

ODP.NET, Managed Driver and ODP.NET do not support Safe Type Mapping.

This section includes the following topics:

- [Comparison Between Oracle Data Types and .NET Types](#)
- [SafeMapping Property](#)

Comparison Between Oracle Data Types and .NET Types

The following sections provide more details about the differences between the Oracle data types and the corresponding .NET types. In general, the Oracle data types allow a greater degree of precision than the .NET types do.

Oracle NUMBER Type to .NET Decimal Type

The Oracle data type NUMBER can hold up to 38 precision, and the .NET Decimal type can hold up to 28 precision. If a NUMBER data type that has more than 28 precision is retrieved into a .NET Decimal type, it loses precision.

[Table 3-39](#) lists the maximum and minimum values for Oracle NUMBER and .NET Decimal types.

Table 3-39 Oracle NUMBER to .NET Decimal Comparisons

Value Limits	Oracle NUMBER	.NET Decimal
Maximum	9.99999999999999999999999999999999 99 e125	79,228,162,514,264,337,593,543,950,335
Minimum	-9.99999999999999999999999999999999 999 e125	-79,228,162,514,264,337,593,543,950,335

Oracle Date Type to .NET DateTime Type

The Oracle data type DATE can represent dates in BC whereas the .NET DateTime type cannot. If a DATE that goes to BC get retrieved into a .NET DateTime type, it loses data.

[Table 3-40](#) lists the maximum and minimum values for Oracle Date and .NET DateTime types.

Table 3-40 Oracle Date to .NET DateTime Comparisons

Value Limits	Oracle Date	.NET DateTime
Maximum	Dec 31, 9999 AD	Dec 31, 9999 AD 23:59:59.9999999
Minimum	Jan 1, 4712 BC	Jan 1, 0001 AD 00:00:00.0000000

Oracle TimeStamp Type to .NET DateTime Type

Similar to the DATE data type, the Oracle TimeStamp data type can represent a date in BC, and a .NET DateTime type cannot. If a TimeStamp that goes to BC is retrieved into a .NET DateTime type, it loses data. The Oracle TimeStamp type can represent values in units of e-9; the .NET DateTime type can represent only values in units of e-7. The Oracle TimeStamp with time zone data type can store time zone information, and the .NET DateTime type cannot.

[Table 3-41](#) lists the maximum and minimum values for Oracle TimeStamp and .NET DateTime types.

Table 3-41 Oracle TimeStamp to .NET DateTime Comparisons

Value Limits	Oracle TimeStamp	.NET DateTime
Maximum	Dec 31, 9999 AD 23:59:59.999999999	Dec 31, 9999 AD 23:59:59.9999999
Minimum	Jan 1, 4712 BC 00:00:00.000000000	Jan 1, 0001 AD 00:00:00.0000000

Oracle INTERVAL DAY TO SECOND to .NET TimeSpan

The Oracle data type INTERVAL DAY TO SECOND can hold up to 9 precision, and the .NET TimeSpan type can hold up to 7 precision. If an INTERVAL DAY TO SECOND data type that has more than 7 precision is retrieved into a .NET TimeSpan type, it loses precision. The Oracle INTERVAL DAY TO SECOND type can represent values in units of e-9, and the .NET TimeSpan type can represent only values in units of e-7.

[Table 3-42](#) lists the maximum and minimum values for Oracle INTERVAL DAY TO SECOND and .NET DateTime types.

Table 3-42 Oracle INTERVAL DAY TO SECOND to .NET TimeSpan Comparisons

Value Limits	Oracle INTERVAL DAY TO SECOND	.NET TimeSpan
Maximum	+999999999 23:59:59.999999999	+10675199 02:48:05.4775807
Minimum	-999999999 23:59:59.999999999	-10675199 02:48:05.4775808

SafeMapping Property

The OracleDataAdapter Safe Type Mapping feature prevents data loss when populating Oracle data for any of these types into a .NET DataSet. By setting the SafeMapping property appropriately, these types can be safely represented in the DataSet, as either of the following:

- .NET byte[] in Oracle format
- .NET String

By default, Safe Type Mapping is disabled.

Using Safe Type Mapping

To use the Safe Type Mapping feature, the OracleDataAdapter.SafeMapping property must be set with a hash table of key-value pairs. The key-value pairs must map database table column names (of type string) to a .NET type (of type Type). ODP.NET supports Safe Type Mapping to byte[] and String types. Any other type mapping causes an exception.

In situations where the column names are not known at design time, an asterisk ("*") can be used to map all occurrences of database types to a safe .NET type. If both the valid column name and the asterisk are present, the column name is used.

Note

- Database table column names are case-sensitive.
- Column names in the hash table that correspond to invalid column names are ignored.

Safe Type Mapping as a string is more readable without further conversion. Converting certain Oracle data types to a string requires extra conversion, which can be slower than converting it to a byte[]. Conversion of .NET strings back to ODP.NET types relies on the formatting information of the session.

SafeTyping Example

```
// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;

class SafeMappingSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";

        // In this SELECT statement, EMPNO, HIREDATE and SALARY must be
        // preserved using safe type mapping.
        string cmdstr = "SELECT EMPNO, ENAME, HIREDATE, SAL FROM EMP";

        // Create the adapter with the selectCommand txt and the connection string
        OracleDataAdapter adapter = new OracleDataAdapter(cmdstr, constr);

        // Get the connection from the adapter
        OracleConnection connection = adapter.SelectCommand.Connection;

        // Create the safe type mapping for the adapter
        // which can safely map column data to byte arrays, where
        // applicable. By executing the following statement, EMPNO, HIREDATE AND
        // SALARY columns will be mapped to byte[]
        adapter.SafeMapping.Add("*", typeof(byte[]));

        // Map HIREDATE to a string
        // If the column name in the EMP table is case-sensitive,
        // the safe type mapping column name must be case-sensitive.
        adapter.SafeMapping.Add("HIREDATE", typeof(string));

        // Map EMPNO to a string
        // If the column name in the EMP table is case-sensitive,
        // the safe type mapping column name must also be case-sensitive.
        adapter.SafeMapping.Add("EMPNO", typeof(string));
        adapter.SafeMapping.Add("SAL", typeof(string));

        // Create and fill the DataSet using the EMP
        DataSet dataset = new DataSet();
        adapter.Fill(dataset, "EMP");

        // Get the EMP table from the dataset
        DataTable table = dataset.Tables["EMP"];
```

```
// Get the first row from the EMP table
DataRow row = table.Rows[0];

// Print out the row info
Console.WriteLine("EMPNO Column: type = " + row["EMPNO"].GetType() +
    "; value = " + row["EMPNO"]);
Console.WriteLine("ENAME Column: type = " + row["ENAME"].GetType() +
    "; value = " + row["ENAME"]);
Console.WriteLine("HIREDATE Column: type = " + row["HIREDATE"].GetType() +
    "; value = " + row["HIREDATE"]);
Console.WriteLine("SAL Column: type = " + row["SAL"].GetType() +
    "; value = " + row["SAL"]);
    }
}
```

① See Also

["SafeMapping"](#)

OracleDataAdapter Requery Property

The `OracleDataAdapter Requery` property controls whether or not queries are reexecuted for `OracleDataAdapter Fill` calls after the initial `Fill` call.

The `OracleDataAdapter Fill` method allows appending or refreshing data in the `DataSet`. When appending the `DataSet` using the same query with subsequent `Fill` calls, reexecuting the query may not be desirable.

When the `Requery` property is set to `true`, each subsequent `Fill` call reexecutes the query and fills the `DataSet`. This is an expensive operation, and if the reexecution is not required, set `Requery` to `false`. If any of the `SelectCommand` properties or associated parameters must be changed, `Requery` must be set to `true`.

When the `Requery` property is set to `false`, the `DataSet` has all the data as a snapshot at a particular time. The query is executed only for the first `Fill` call; subsequent `Fill` calls fetch the data from a cursor opened with the first execution of the query. This feature is supported only for forward-only fetches. `Fill` calls that try to fetch rows before the last fetched row raise an exception. The connection used for the first `Fill` call must be available for subsequent `Fill` calls.

When filling a `DataSet` with an `OracleRefCursor` object, the `Requery` property can be used in a similar manner. When the `Requery` property is set to `false`, both the connection used for the first `Fill` call and the `OracleRefCursor` object must be available for the subsequent `Fill` calls.

① See Also

- ["Requery"](#)
- ["SelectCommand"](#)

Guaranteeing Uniqueness in Updating DataSet to Database

This section describes how the `OracleDataAdapter` object configures the `PrimaryKey` and `Constraints` properties of the `DataTable` object which guarantee uniqueness when the `OracleCommandBuilder` object is updating `DataSet` changes to the database.

Using the `OracleCommandBuilder` object to dynamically generate DML statements to be executed against the database is one of the ways to reconcile changes made in a single `DataTable` object with the database.

In this process, the `OracleCommandBuilder` object must not be allowed to generate DML statements that may affect (update or delete) more than a single row in the database when reconciling a single `DataRow` change. Otherwise the `OracleCommandBuilder` could corrupt data in the database.

To guarantee that each `DataRow` object change affects only a single row, there must be a set of `DataColumn` objects in the `DataTable` for which all rows in the `DataTable` have a unique set of values. The set of `DataColumn` objects indicated by the properties `DataTable.PrimaryKey` and `DataTable.Constraints` meets this requirement. The `OracleCommandBuilder` object determines uniqueness in the `DataTable` by checking if the `DataTable.PrimaryKey` is not a null value or if there exists a `UniqueConstraint` object in the `DataTable.Constraints` collection.

This discussion first explains what constitutes uniqueness in `DataRow` objects and then explains how to maintain that uniqueness while updating, through the `DataTable` property configuration.

This section includes the following topics:

- [What Constitutes Uniqueness in DataRow Objects?](#)
- [Configuring PrimaryKey and Constraints Properties](#)
- [Updating Without PrimaryKey and Constraints Configuration](#)

What Constitutes Uniqueness in DataRow Objects?

This section describes the minimal conditions that must be met to guarantee uniqueness of `DataRow` objects. The condition of uniqueness must be guaranteed before the `DataTable.PrimaryKey` and `DataTable.Constraints` properties can be configured, as described in the next section.

Uniqueness is guaranteed in a `DataTable` object if any one of the following is true:

- All the columns of the primary key are in the select list of the `OracleDataAdapter.SelectCommand` property.
- All the columns of a unique constraint are in the select list of the `OracleDataAdapter.SelectCommand` property, with at least one involved column having a NOT NULL constraint defined on it.
- All the columns of a unique index are in the select list of the `OracleDataAdapter.SelectCommand` property, with at least one of the involved columns having a NOT NULL constraint defined on it.
- A ROWID is present in the select list of the `OracleDataAdapter.SelectCommand` property.

Note

A set of columns, on which a unique constraint has been defined or a unique index has been created, requires at least one column that cannot be null for the following reason: if all the columns of the column set can be null, then multiple rows could exist that have a NULL value for each column in the column set. This would violate the uniqueness condition that each row has a unique set of values for the column set.

Configuring PrimaryKey and Constraints Properties

If the minimal conditions described in "[What Constitutes Uniqueness in DataRow Objects?](#)" are met, then the `DataTable.PrimaryKey` or `DataTable.Constraints` properties can be set.

After these properties are set, the `OracleCommandBuilder` object can determine uniqueness in the `DataTable` by checking the `DataTable.PrimaryKey` property or the presence of a `UniqueConstraint` object in the `DataTable.Constraints` collection. Once uniqueness is determined, the `OracleCommandBuilder` object can safely generate DML statements to update the database.

The `OracleDataAdapter.FillSchema` method attempts to set these properties according to this order of priority:

1. If the primary key is returned in the select list, it is set as the `DataTable.PrimaryKey` property.
2. If a set of columns that meets the following criteria is returned in the select list, it is set as the `DataTable.PrimaryKey` property.

Criteria: The set of columns has a unique constraint defined on it or a unique index created on it, with each column having a NOT NULL constraint defined on it.

3. If a set of columns that meets the following criteria is returned in the select list, a `UniqueConstraint` object is added to the `DataTable.Constraints` collection, but the `DataTable.PrimaryKey` property is not set.

Criteria: The set of columns has a unique constraint defined on it or a unique index created on it, with at least one column having a NOT NULL constraint defined on it.

4. If a ROWID is part of the select list, it is set as the `DataTable.PrimaryKey` property.

Additionally, the `OracleDataAdapter.FillSchema` method performs as follows:

- Setting the `DataTable.PrimaryKey` property implicitly creates a `UniqueConstraint` object.
- If a column is part of the `DataTable.PrimaryKey` property or the `UniqueConstraint` object, or both, it will be repeated for each occurrence of the column in the select list.

Updating Without PrimaryKey and Constraints Configuration

If the `DataTable.PrimaryKey` or `Constraints` properties have not been configured, for example, if the application has not called the `OracleDataAdapter.FillSchema` method, the `OracleCommandBuilder` object directly checks the select list of the `OracleDataAdapter.SelectCommand` property to determine if it guarantees uniqueness in the `DataTable`. However this check results in a database round-trip to retrieve the metadata for the `SELECT` statement of the `OracleDataAdapter.SelectCommand`.

Note that `OracleCommandBuilder` object cannot update a `DataTable` created from PL/SQL statements because they do not return any key information in their metadata.

Globalization Support

ODP.NET globalization support enables applications to manipulate culture-sensitive data appropriately. This feature ensures proper string format, date, time, monetary, numeric, sort order, and calendar conventions depending on the Oracle globalization settings.

Note

- ODP.NET, Managed Driver and ODP.NET Core are not `NLS_LANG` sensitive. They are only .NET locale sensitive.
- ODP.NET, Managed Driver and ODP.NET Core do not support thread-based globalization.

See Also

["OracleGlobalization Class"](#)

This section includes the following:

- [Globalization Settings](#)
- [Globalization-Sensitive Operations](#)

Globalization Settings

An `OracleGlobalization` object can be used to represent the following:

- [Client Globalization Settings](#)
- [Session Globalization Settings](#)
- [Thread-Based Globalization Settings](#)

Client Globalization Settings

Client globalization settings are derived from the Oracle globalization setting (`NLS_LANG`) in the Windows registry of the local computer. The client globalization parameter settings are read-only and remain constant throughout the lifetime of the application. These settings can be obtained by calling the `OracleGlobalization.GetClientInfo` static method.

The following example retrieves the client globalization settings:

```
// C#  
  
using System;  
using Oracle.DataAccess.Client;  
  
class ClientGlobalizationSample  
{  
    static void Main()
```

```

    {
        OracleGlobalization ClientGlob = OracleGlobalization.GetClientInfo();

        Console.WriteLine("Client machine language: " + ClientGlob.Language);
        Console.WriteLine("Client charset: " + ClientGlob.ClientCharacterSet);
    }
}

```

The properties of the `OracleGlobalization` object provide the Oracle globalization value settings.

Session Globalization Settings

Session globalization parameters are initially identical to client globalization settings. Unlike client settings, session globalization settings can be updated. However, they can be obtained only after establishing a connection against the database. The session globalization settings can be obtained by calling the `GetSessionInfo` method on the `OracleConnection` object. Invoking this method returns an instance of an `OracleGlobalization` class whose properties represent the globalization settings of the session.

When the `OracleConnection` object establishes a connection, it implicitly opens a session whose globalization parameters are initialized with those values specified by the client computer's Oracle globalization (or (NLS)) registry settings. The session settings can be updated and can change during its lifetime.

The following example changes the date format setting on the session:

```

// C#

using System;
using Oracle.DataAccess.Client;

class SessionGlobalizationSample
{
    static void Main()
    {
        OracleConnection con = new OracleConnection();

        con.ConnectionString = "User Id=scott;Password=tiger;Data Source=oracle;";
        con.Open();

        OracleGlobalization SessionGlob = con.GetSessionInfo();

        // SetSessionInfo updates the Session with the new value
        SessionGlob.DateFormat = "YYYY/MM/DD";
        con.SetSessionInfo(SessionGlob);
        Console.WriteLine("Date Format successfully changed for the session");

        // Close and Dispose OracleConnection object
        con.Close();
        con.Dispose();
    }
}

```

Thread-Based Globalization Settings

Thread-based globalization parameter settings are specific to each thread. Initially, these settings are identical to the client globalization parameters, but they can be changed as specified by the application. When ODP.NET Types are converted to and from strings, the thread-based globalization parameters are used, if applicable.

Thread-based globalization parameter settings are obtained by invoking the `GetThreadInfo` static method of the `OracleGlobalization` class. The `SetThreadInfo` static method of the `OracleGlobalization` class can be called to set the thread's globalization settings.

ODP.NET classes and structures rely solely on the `OracleGlobalization` settings when manipulating culture-sensitive data. They do not use .NET thread culture information. If the application uses only .NET types, `OracleGlobalization` settings have no effect. However, when conversions are made between ODP.NET types and .NET types, `OracleGlobalization` settings are used where applicable.

① Note

Changes to the `System.Threading.Thread.CurrentCulture` property do not impact the `OracleGlobalization` settings of the thread or the session, or the reverse.

The following example shows how the thread's globalization settings are used by the ODP.NET Types:

```
// C#

using System;
using Oracle.DataAccess.Types;
using Oracle.DataAccess.Client;

class ThreadBasedGlobalizationSample
{
    static void Main(string[] args)
    {
        // Set the thread's DateFormat for the OracleDate constructor
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.DateFormat = "YYYY-MON-DD";
        OracleGlobalization.SetThreadInfo(info);

        // construct OracleDate from a string using the DateFormat specified.
        OracleDate date = new OracleDate("1999-DEC-01");

        // Set a different DateFormat for the thread
        info.DateFormat = "MM/DD/YYYY";
        OracleGlobalization.SetThreadInfo(info);

        // Print "12/01/1999"
        Console.WriteLine(date.ToString());
    }
}
```

The `OracleGlobalization` object validates property changes made to it. If an invalid value is used to set a property, an exception is thrown. Note that changes made to the `Territory` and `Language` properties change other properties of the `OracleGlobalization` object implicitly.

① See Also

Oracle Database Globalization Support Guide for more information on the properties affected by `Territory` and `Language` globalization settings

Globalization-Sensitive Operations

This section lists ODP.NET types and operations that are dependent on or sensitive to globalization settings.

Operations Dependent on Client Computer's Globalization Settings

The `OracleString` structure depends on the `OracleGlobalization` settings of the client computer. The client character set of the local computer is used when it converts a Unicode string to a `byte[]` in the `GetNonUnicode` method and when it converts a `byte[]` of ANSI characters to Unicode in the `OracleString` constructor that accepts a `byte[]`.

Operations Dependent on Thread Globalization Settings

The thread globalization settings are used by ODP.NET types whenever they are converted to and from .NET string types, where applicable. Specific thread globalization settings are used in most cases, depending on the ODP.NET type, by the following:

- The `ToString` method
- The `Parse` static method
- Constructors that accept .NET string data
- Conversion operators to and from .NET strings

For example, the `OracleDate` type uses the `DateFormat` property of the thread globalization settings when the `ToString` method is invoked on it. This returns a `DATE` as a string in the format specified by the thread's settings.

The thread globalization settings also affect data that is retrieved into the `DataSet` as a string using Safe Type Mapping. If the type is format-sensitive, the strings are always in the format specified by the thread globalization settings.

For example, `INTERVAL DAY TO SECOND` data is not affected by thread settings because no format is applicable for this type. However, the `DateFormat` and `NumericCharacters` properties can impact the string representation of `DATE` and `NUMBER` types, respectively, when they are retrieved as strings into the `DataSet` through Safe Type Mapping.

See Also

- The remarks in [OracleGlobalization Class](#) for more details on the ODP.NET type methods that convert between ODP.NET types and .NET string types, and to identify which thread globalization settings are used for that particular method.
- [OracleDataAdapter Safe Type Mapping](#)
- [Oracle Data Provider for .NET Types Structures](#)

Operations Sensitive to Session Globalization Parameters

Session globalization settings affect any data that is retrieved from or sent to the database as a string.

For example, if a DATE column is selected with the TO_CHAR function applied on it, the DATE column data will be a string in the date format specified by the DateFormat property of the session globalization settings. Transmitting data in the other direction, the string data that is to be inserted into the DATE column, must be in the format specified by the DateFormat property of the session globalization settings.

ODP.NET Driver Globalization Differences

Default Session Time Zone

ODP.NET drivers set the default session time zone differently. While the session time zone for unmanaged ODP.NET uses an hour offset, managed ODP.NET and ODP.NET Core use the region identifier for setting their respective session time zone. As a result, managed and Core ODP.NET is sensitive to daylight savings in scenarios where the timestamp LTZ values have to be converted from or to the session time zone.

There are two methods to resolve this difference if needed. For ODP.NET, Unmanaged Driver, the application explicitly sets the region identifier with the environment variable ORA_SDTZ. For example, set ORA_SDTZ = <Region ID>. If ORA_SDTZ variable is set, Oracle Client considers this value as the session time zone. The second method is to execute an alter session command to set the session time zone property to the region identifier.

.NET Culture Info

As noted earlier, ODP.NET, Managed Driver and ODP.NET Core do not support thread-based globalization. ODP.NET, Unmanaged Driver does. As a result, managed ODP.NET and ODP.NET Core strings are insensitive to .NET culture information. As such, invoking the ToString() method on date/time related Oracle provider types (i.e. OracleDate, OracleTimestamp, etc.) will generate a string in the format of MM/dd/yyyy HH:mm:ss[.ffffffffff] [timezone]. Whether the outputted string contains a fractional second and/or the timezone will depend on the Oracle provider type being used.

Data Use Case Domains

Oracle AI Database 26ai introduces Data Use Case Domains, a dictionary object that belongs to a schema and encapsulates a set of optional properties and constraints for common values, such as credit card numbers or email addresses. After you define a use case domain, you can define table columns to be associated with that domain, thereby explicitly applying the domain's optional properties and constraints to those columns.

With Data Use Case Domains, you can define how you intend to use data centrally. They make it easier to ensure you handle values consistently across applications and improve data quality.

All ODP.NET provider types support using Data Use Case Domains, as well as retrieving Data Use Case Domains schema and name information using `OracleDataReader.GetSchemaTable()`. The `SchemaTable` has two columns, `DomainName` and `DomainSchema`, that stores this information.

Annotations

Oracle Database annotations, a new Oracle AI Database 26ai feature, allow storing database object metadata. They comprise a name, commonly called a key, and an optional value. Both are freeform text fields. A schema object can have more than one annotation.

ODP.NET Core, managed, and unmanaged drivers all support Oracle Database annotations in table and view columns. ODP.NET materializes them as JSON formatted strings in those schema objects.

ODP.NET can retrieve database annotations using `OracleDataReader.GetSchemaTable`. The `SchemaTable` has one column, `Annotations`, that stores this information as JSON formatted string.

OpenTelemetry

Modern computing has seen more distributed systems scaled and more services added, such as with microservices and serverless applications. With this growth, service ownership across the system is allocated to different individuals, or even organizations. It is increasingly difficult to observe how these services depend on each and affect other services without a unified observability framework. This becomes clear after a deployment or during an outage when issue identification requires speed and accuracy.

OpenTelemetry is such a unified observability framework. It is a popular open-source observability framework for instrumenting, generating, collecting, and exporting telemetry data. It provides a common specification and protocol so that multiple services can furnish a unified version of traces, metrics, and logs. OpenTelemetry relies on components that participate in the observability model to be instrumented, which means the code emits traces, metrics, and logs.

Numerous managed ODP.NET and ODP.NET Core APIs have been instrumented to support OpenTelemetry observability and standards. Developers and operators can then customize the ODP.NET metrics collected.

ODP.NET OpenTelemetry Traces

Traces record the request paths taken either by an application or end user as they propagate through a multi-service architecture. Traces in OpenTelemetry are defined implicitly by their spans. A trace can be thought of as a directed acyclic graph of spans where the edges between spans are defined as parent/child relationships.

A span, or activity, represents an operation within a transaction. Each span encapsulates the following state:

- Display name
- ODP.NET NuGet version
- Start timestamp and duration
- Attributes – list of key-value pairs
- Events – a tuple (timestamp, name, attributes) and can number from zero or more
- The parent span identifier
- Links to causally related spans through `SpanContext`, which can number from zero or more

ODP.NET spans become child spans if the application creates a (parent) span before calling ODP.NET instrumented APIs.

ODP.NET OpenTelemetry publishes the following activity tags, or attributes.

Table 3-43 ODP.NET OpenTelemetry activity tags

Attribute Name	Description
Activity.DisplayName	Activity display name or the name of the operation being instrumented.
db.name	Database being accessed by the command. The attribute value is the same as the OracleConnection DatabaseName property.
db.odp.connection.id	Gives each connection a unique identifier for diagnostic purposes. This identifier can be prefixed with a custom application generated string by setting the ConnectionIdPrefix property in OracleConnection or OracleCommand.
db.odp.roundtrip.count	Number of database server roundtrips the ODP.NET method makes.
db.odp.roundtrip.duration	Database server roundtrip time, which includes the time needed for ODP.NET processing, network sending and receiving, and database server processing.
db.odp.rows_affected	Row count affected by the SQL DML (INSERT, UPDATE, and DELETE only) when using ExecuteNonQuery*, ExecuteReader*, or ExecuteScalar* methods. Does not return a value for stored procedures and anonymous PL/SQL.
db.odp.sql_id	Oracle database SQL statement identifier value, equivalent to SQL_ID
db.odp.user.statement	Redacted ODP.NET user-supplied command text which can differ from the database statement being executed, such as the case with stored procedures and XML commands. This tag is instrumented only when the following two conditions are met: • The internally executed statement is different from the user-supplied command text. • setDBStatementForText or SetDbStatementForStoredProcedure is set to true, depending on which is applicable, just like db.statement tag.
db.response.returned_rows	Number of rows fetched per database round trip. This tag is active for SELECT statements only.
db.statement	Redacted database statement being executed
db.system	Database name. Oracle is the attribute value for Oracle databases.
db.user	Database user
exception.message	Exception message
exception.stacktrace	Exception stack trace
exception.type	Exception type
otel.status_code	Status code of the trace
otel.status_description	Status description of the trace

Table 3-43 (Cont.) ODP.NET OpenTelemetry activity tags

Attribute Name	Description
server.address	Name of the database host
server.port	Server port number

ODP.NET OpenTelemetry Trace Sample Exporter Visualization:

```

Activity.TraceId:          29399ee7598dea88defd7c5421bc41d4
Activity.SpanId:           e1878b5fc4435f23
Activity.TraceFlags:       Recorded
Activity.ParentSpanId:     8487a50b9ace7808
Activity.ActivitySourceName: Oracle.ManagedDataAccess.Core
Activity.ActivitySourceVersion: 23.6.0
Activity.DisplayName:      Connect MYHOSTNAME:1521:dbview
Activity.Kind:             Client
Activity.StartTime:        2024-09-19T07:26:16.4615973Z
Activity.Duration:         00:00:00.4114888
Activity.Tags:
    db.system: oracle
    db.user: scott
    db.name: dbview
    server.address: MYHOSTNAME
    server.port: 1521

```

```

StatusCode: Ok
Resource associated with Activity:
    service.name: DemoApp
    service.version: 1.0.0
    service.instance.id: 312ec47d-c0a1-4df3-b59f-f38aa225c39d
    telemetry.sdk.name: opentelemetry
    telemetry.sdk.language: dotnet
    telemetry.sdk.version: 1.9.0

```

```

Activity.TraceId:          29399ee7598dea88defd7c5421bc41d4
Activity.SpanId:           8487a50b9ace7808
Activity.TraceFlags:       Recorded
Activity.ParentSpanId:     95b69d7844ae0dd3
Activity.ActivitySourceName: Oracle.ManagedDataAccess.Core
Activity.ActivitySourceVersion: 23.6.0
Activity.DisplayName:      Open MYHOSTNAME:1521:dbview
Activity.Kind:             Client
Activity.StartTime:        2024-09-19T07:26:16.3584252Z
Activity.Duration:         00:00:00.5781012

```

```

Activity.Tags:
    db.system: oracle
    db.user: scott
    db.name: dbview
    server.address: MYHOSTNAME
    server.port: 1521
StatusCode: Ok
Resource associated with Activity:
    service.name: DemoApp

```

```
service.version: 1.0.0
service.instance.id: 312ec47d-c0a1-4df3-b59f-f38aa225c39d
telemetry.sdk.name: opentelemetry
telemetry.sdk.language: dotnet
telemetry.sdk.version: 1.9.0

Activity.TraceId:          29399ee7598dea88defd7c5421bc41d4
Activity.SpanId:           89b3cd0b7b2d2f35
Activity.TraceFlags:       Recorded
Activity.ParentSpanId:     e0cf4916dc1e4328
Activity.ActivitySourceName: Oracle.ManagedDataAccess.Core
Activity.ActivitySourceVersion: 23.6.0
Activity.DisplayName:      SendExecuteRequest MYHOSTNAME:1521:dbview
Activity.Kind:             Client
Activity.StartTime:        2024-09-19T07:26:17.0031962Z
Activity.Duration:         00:00:00.0492725
Activity.Tags:
  db.system: oracle
  server.address: MYHOSTNAME
  server.port: 1521
  db.name: dbview
  db.user: scott
  db.statement: update Blogs set Blog_Name=? where Blog_Id=?
  db.odp.sql_id: 71jyh89d1cz9r
StatusCode: Ok
Resource associated with Activity:
  service.name: DemoApp
  service.version: 1.0.0
  service.instance.id: 312ec47d-c0a1-4df3-b59f-f38aa225c39d
  telemetry.sdk.name: opentelemetry
  telemetry.sdk.language: dotnet
  telemetry.sdk.version: 1.9.0

Activity.TraceId:          29399ee7598dea88defd7c5421bc41d4
Activity.SpanId:           e0cf4916dc1e4328
Activity.TraceFlags:       Recorded
Activity.ParentSpanId:     95b69d7844ae0dd3
Activity.ActivitySourceName: Oracle.ManagedDataAccess.Core
Activity.ActivitySourceVersion: 23.6.0
Activity.DisplayName:      ExecuteNonQuery MYHOSTNAME:1521:dbview
Activity.Kind:             Client
Activity.StartTime:        2024-09-19T07:26:16.9671882Z
Activity.Duration:         00:00:00.1279960
Activity.Tags:
  db.system: oracle
  server.address: MYHOSTNAME
  server.port: 1521
  db.name: dbview
  db.user: scott
  db.statement: update Blogs set Blog_Name=? where Blog_Id=?
  db.odp.rows_affected: 1
  db.odp.sql_id: 71jyh89d1cz9r
StatusCode: Ok
Resource associated with Activity:
  service.name: DemoApp
  service.version: 1.0.0
```

```
service.instance.id: 312ec47d-c0a1-4df3-b59f-f38aa225c39d
telemetry.sdk.name: opentelemetry
telemetry.sdk.language: dotnet
telemetry.sdk.version: 1.9.0

Activity.TraceId:          29399ee7598dea88defd7c5421bc41d4
Activity.SpanId:           8d2ebac56ee5c04f
Activity.TraceFlags:       Recorded
Activity.ParentSpanId:     95b69d7844ae0dd3
Activity.ActivitySourceName: Oracle.ManagedDataAccess.Core
Activity.ActivitySourceVersion: 23.6.0
Activity.DisplayName:      Close MYHOSTNAME:1521:dbview
Activity.Kind:             Client
Activity.StartTime:        2024-09-19T07:26:17.1450407Z
Activity.Duration:         00:00:00.0144354
Activity.Tags:
  db.system: oracle
  server.address: MYHOSTNAME
  server.port: 1521
  db.name: dbview
  db.user: scott
StatusCode: Ok
Resource associated with Activity:
  service.name: DemoApp
  service.version: 1.0.0
  service.instance.id: 312ec47d-c0a1-4df3-b59f-f38aa225c39d
  telemetry.sdk.name: opentelemetry
  telemetry.sdk.language: dotnet
  telemetry.sdk.version: 1.9.0

Activity.TraceId:          29399ee7598dea88defd7c5421bc41d4
Activity.SpanId:           95b69d7844ae0dd3
Activity.TraceFlags:       Recorded
Activity.ActivitySourceName: Test
Activity.DisplayName:      SampleTestActivity
Activity.Kind:             Internal
Activity.StartTime:        2024-09-19T07:26:16.2164456Z
Activity.Duration:         00:00:00.9785065
Resource associated with Activity:
  service.name: DemoApp
  service.version: 1.0.0
  service.instance.id: 312ec47d-c0a1-4df3-b59f-f38aa225c39d
  telemetry.sdk.name: opentelemetry
  telemetry.sdk.language: dotnet
  telemetry.sdk.version: 1.9.0
```

ODP.NET OpenTelemetry supports both dynamic and manual instrumentation. The next section describes how to set up each instrumentation method.

Dynamic and Manual Instrumentation

ODP.NET OpenTelemetry requires an exporter, such Jaeger or Zipkin, to visualize and analyze traces. These exporters are available as NuGet packages and will also include the OpenTelemetry SDK as a NuGet dependency.

ODP.NET OpenTelemetry manual instrumentation has the following requirements:

- Oracle.ManagedDataAccess.OpenTelemetry package available from NuGet Gallery.
- Exporter NuGet package, such as Console or Zipkin, which is used for instrumentation visualization.
- OpenTelemetry SDK, which will be installed automatically with an exporter NuGet package.

Once the ODP.Net OpenTelemetry nuget package is installed, ODP.NET OpenTelemetry is enabled by invoking the `TracerProviderBuilder.AddOracleDataProviderInstrumentation` extension method upon application startup. It accepts various options for configuring OpenTelemetry instrumentation.

Code sample: Enabling ODP.NET OpenTelemetry instrumentation with default options

```
using OpenTelemetry.Trace;

Sdk.CreateTracerProviderBuilder()
    .AddOracleDataProviderInstrumentation() // ODP.NET extension method
    .AddConsoleExporter()
    .Build();
```

Code sample: Enabling ODP.NET OpenTelemetry instrumentation with all options enabled

```
using OpenTelemetry.Trace;

Sdk.CreateTracerProviderBuilder()
    .AddOracleDataProviderInstrumentation(o =>
    {
        o.EnableConnectionLevelAttributes = true;
        o.RecordException = true;
        o.InstrumentOracleDataReaderRead = true;
        o.SetDbStatementForText = true;
    })
    .AddConsoleExporter()
    .Build();
```

After enabling ODP.NET OpenTelemetry, whenever a provider instrumented API is called, OpenTelemetry traces will be generated and sent to the configured exporter.

By default, instrumentation for potentially sensitive data is disabled, such as for SQL statements.

A second manual instrumentation method allows enabling ODP.NET OpenTelemetry without adding the ODP.NET OpenTelemetry NuGet package and without calling the `AddOracleDataProviderInstrumentation()` extension method. It requires adding the `TracerProviderBuilder` call `AddSource("Oracle.ManagedDataAccess.Core")` for ODP.NET Core or `AddSource("Oracle.ManagedDataAccess")` for managed ODP.NET.

ODP.NET OpenTelemetry instrumentation will then be enabled using default options. The defaults must be used as there is no way to modify option values. Changing these values require using ODP.NET OpenTelemetry NuGet package directly.

`OpenTelemetry.Api` NuGet package is installed automatically if the application adds any OpenTelemetry exporter NuGet packages, such as `OpenTelemetry.Exporter.Console` or `OpenTelemetry.Exporter.Zipkin`. `OpenTelemetry.Api` is necessary to call the `AddSource()` method.

Code sample: Enabling ODP.NET OpenTelemetry manual instrumentation via AddSource

```
Sdk.CreateTracerProviderBuilder()  
    .AddSource("Oracle.ManagedDataAccess.Core")  
    .AddConsoleExporter()  
    .Build();
```

Through this instrumentation method, it is possible to enable and disable ODP.NET OpenTelemetry tracing programmatically at runtime dynamically. To enable, set `OracleConfiguration.OpenTelemetryTracing` to `true` (default). To disable, set it to `false`.

`OpenTelemetryTracing` is available as a managed ODP.NET .NET configuration file setting as well. As a .NET config setting, it is read upon app startup only. Changes to this setting's value after startup are ignored.

Automatic Instrumentation

OpenTelemetry automatic instrumentation enables ODP.NET tracing without requiring code changes. Existing .NET apps can turn on tracing without having to recompile.

To use it, add the `OpenTelemetry.AutoInstrumentation 1.7.0` NuGet package or higher to your project. Older package versions require setting environment variables to enable ODP.NET tracing. For example, the following environment variable settings turn on ODP.NET Core tracing using the console exporter.

```
OTEL_DOTNET_AUTO_TRACES_ADDITIONAL_SOURCES=Oracle.ManagedDataAccess.Core  
OTEL_DOTNET_AUTO_TRACES_CONSOLE_EXPORTER_ENABLED=true
```

ODP.NET automatic instrumentation is on by default when using a newer `OpenTelemetry.AutoInstrumentation` NuGet package.

Instrumented ODP.NET APIs

The list of managed ODP.NET and ODP.NET Core APIs that are instrumented to support OpenTelemetry is as follows. Every ODP.NET database round trip originating from one of these APIs is instrumented as well.

- `OracleCommand`
 - `ExecuteNonQuery()`
 - All `ExecuteNonQueryAsync` overloads
 - All `ExecuteReader` overloads
 - All `ExecuteReaderAsync` overloads
 - `ExecuteScalar()`
 - All `ExecuteScalarAsync` overloads
 - `ExecuteStream()`
 - `ExecuteToStream(Stream)`
 - `ExecuteXmlReader()`
 - All `ExecuteXmlReaderAsync` overloads

- `OracleConnection`
 - `Close()`
 - `Open()`
 - All `OpenAsync()` overloads
- `OracleDataAdapter`
 - All `Fill` overloads
- `OracleDataReader`
 - `Read()`
 - All `ReadAsync` overloads

See Also

[Oracle Data Provider for .NET OpenTelemetry and Metrics Classes](#)

[OpenTelemetry](#) documentation

.NET Metrics

.NET Metrics are application numerical measurements collected at regular time intervals for the purposes of monitoring and alerting about application health. Typically, these metrics are delivered to a monitoring system. In an ODP.NET setting, metrics can monitor connection statistics, such as number of ODP.NET hard connections to the database, number of active connections, or number of free connections.

Starting with version 23.3.2, ODP.NET Core and managed ODP.NET support .NET Metrics. ODP.NET Core supports metrics on .NET (Core) for all operating systems ODP.NET Core is certified with. To use .NET Metrics, ODP.NET projects must include the NuGet package, `System.Diagnostics.DiagnosticSource` 6.0.0 or higher. This package includes the `System.Diagnostics.Metrics` APIs and allows ODP.NET code to be instrumented for rich data payload logging for consumption within the same process.

For metrics instrumentation, developers will associate the metric name `Oracle.ManagedDataAccess.Core` for ODP.NET Core or `Oracle.ManagedDataAccess` for managed ODP.NET to subscribe to and monitor ODP.NET measurements, such as for the `MeterListener` `MeterName` property.

For metrics collection, ODP.NET metrics is tightly integrated with .NET OpenTelemetry. This means that ODP.NET metrics can be published to and analyzed by the rich and expansive toolsets integrated with OpenTelemetry, such as Grafana and Prometheus. Developers can also use common .NET SDK tools, such as `dotnet-counters`. For example, to live monitor all metrics for all ODP.NET Core processes, execute the command:

```
dotnet-counters monitor "Oracle.ManagedDataAccess.Core"
```

You may indicate a specific process and metric to monitor:

```
dotnet-counters monitor -n <process name> <metrics source name>
```

You may identify the process by its identifier:

```
dotnet-counters monitor <metrics source name> -p <process id>
```

Unlike ODP.NET performance counters, there is no need to enable metrics explicitly. ODP.NET metrics counters instances are always available in-memory after an application starts. However, these metrics values are only published for active subscribers.

Note

.NET Metrics cannot be consumed by Windows Performance Monitor.

ODP.NET can flexibly collect and distinguish connection metrics at multiple granularity levels, including the:

- AppDomain level
- Connection string or pool manager level
- Database instance level

ODP.NET supports the following metric measurements and levels.

Table 3-44 Metric Counters and Levels

Metrics Counters	Description	Levels Supported
HardConnects	Number of sessions established with database	All
HardDisconnects	Number of sessions severed from database	All
SoftConnects	Number of soft connects	All
SoftDisconnects	Number of soft disconnects	All
NumberOfActiveConnectionPools	Number of active connection pools, which are pools with at least one connection	All
NumberOfInactiveConnectionPools	Number of inactive connection pools, which are pools with zero connections	All
NumberOfActiveConnections	Number of connections in use, also known as checked out connections	All
NumberOfFreeConnections	Number of connections available for use in connection pool, also known as queued connections	All
NumberOfPooledConnections	Number of pooled connections, which are the total of checked out and queued connections	All
NumberOfNonPooledConnections	Number of non-pooled active connections	All
NumberOfReclaimedConnections	Number of implicitly garbage-collected connections	All
NumberOfActiveHardConnections	Number of opened database connections	All
NumberOfActiveSoftConnections	Number of connections checked out from the pool	All

Table 3-44 (Cont.) Metric Counters and Levels

Metrics Counters	Description	Levels Supported
NumberOfActiveConnectionPoolGroups	Number of pool groups (i.e., number of unique connection strings) with at least one connection	AppDomain level only
NumberOfInactiveConnectionPoolGroups	Number of pool groups (i.e., number of unique connection strings) with zero connections	AppDomain level only

.NET uses long metric level identifiers that are not easily human readable. They consist of a string with the application name, process id, and connection string. To make identifying the metric level easier, ODP.NET developers can customize the connection string identifier name in the `OraclePoolNameCollection.PoolNames` property. Managed ODP.NET developers also have the option to use the `<connectionPools>` section of the .NET configuration file. Developers will be able to configure a different pool name for each unique connection string.

Each cluster of Oracle Real Application Cluster (RAC) instances will use the same pool name configuration since all the instances in a cluster have an identical connection string.

Debug Tracing

All ODP.NET providers include debug tracing support, which allows logging of all the ODP.NET activities into a trace file. Different levels of tracing are available.

Tracing can record the following information and more:

- Entry and exit information for the ODP.NET public methods
- User-provided SQL statements as well as SQL statements modified by the provider
- Connection pooling statistics such as enlistment and delistment
- Thread ID (entry and exit)
- HA Events and Load Balancing information
- Distributed Transactions
- Self-tuning information
- User-mode dumps upon unmanaged exceptions

The following properties enable and configure tracing for all ODP.NET providers:

- `TraceLevel` – specifies which ODP.NET details to write to the trace file
- `TraceFileLocation` – specifies the directory to write the trace to
- `TraceFileMaxSize` – specifies the maximum file size of each trace file
- `TraceOption` – specifies whether to use a separate trace file for each thread

By default, tracing is turned off (`TraceLevel = 0`). To enable tracing, set `TraceLevel` to the trace level you wish.

Managed ODP.NET and ODP.NET Core have the following trace levels:

- 1 = public APIs
- 2 = private APIs

- 4 = network APIs/data
- 8 = disables writing SQL statements and network packet contents

These values can be bitwise ORed. To enable all traces, set `TraceLevel` to 7.

Unmanaged ODP.NET has the following trace levels:

- 1 = Entry, exit, and SQL statement information
- 2 = Connection pooling statistics
- 4 = Distributed transactions (enlistment and delistment)
- 8 = User-mode dump creation upon unmanaged exception
- 16 = HA Event Information
- 32 = Load Balancing Information
- 64 = Self Tuning Information

These values can be bitwise ORed. To enable all traces, set `TraceLevel` to 127.

`TraceFileLocation` writes to a temporary operating system folder if no value is set.

- ODP.NET Core
 - Windows: *<Windows user temporary folder>\ODP.NET\core\trace*
 - Linux: *<current user temporary folder>/ODP.NET/core/trace*
 - macOS: *<macOS temporary folder>\ODP.NET\core\trace*
- ODP.NET, Managed Driver: *<Windows user temporary folder>\ODP.NET\managed\trace*
- ODP.NET, Unmanaged Driver: *<Windows user temporary folder>\ODP.NET\unmanaged\trace*

The Windows user temporary folder is determined by your local Windows settings, such as your Windows TMP or TEMP environment variable. Typically, it can be `C:\temp` or `C:\Users\<user name>\AppData\Local\Temp`.

`TraceFileMaxSize` defaults to 100 MB if no value is set. Administrators can use this setting to keep trace files down to a manageable size.

`TraceOption` will write all threads to a single trace file at a time by default (`TraceOption=0`).

ODP.NET will create an entry in the Windows event log where the trace was created anytime it creates a new trace file. An event log entry will be added every time in cases when `TraceFileLocation` changes, trace file size exceeds `TraceFileMaxSize`, `TraceLevel` changes, and `TraceOption` changes. In each of these cases, ODP.NET creates a new file and starts writing traces in the newly created file. ODP.NET Core does not support writing to the Windows event log.

To enable ODP.NET tracing, these trace properties must be set appropriately either in the `OracleConfiguration` class (all providers), in the .NET configuration file (managed and unmanaged ODP.NET), or in the Windows Registry (unmanaged ODP.NET).

ODP.NET begins writing to the trace file only after the `OracleConnection.Open()` call occurs.

Dynamic Tracing

Starting with ODAC 19c, applications can change ODP.NET `OracleConfiguration` class trace settings during runtime. Most commonly, this feature would permit enabling and disabling tracing dynamically while applications continue to run. Dynamic tracing is useful for errors that

occur after applications are run for a considerable time. Customers can avoid collecting extremely large trace files and Oracle Support will find it easier to diagnose problems from more targeted trace files.

ODP.NET Core, managed, and unmanaged all support dynamic tracing.

To enable tracing at runtime, set `OracleConfiguration.TraceLevel` to the trace level desired. To disable tracing dynamically, set `OracleConfiguration.TraceLevel` to zero.

Only `OracleConfiguration` tracing properties can alter runtime tracing behavior. Windows Registry and .NET configuration file settings are only read when an application starts up.

As a general matter, tracing is not recommended for production use because writing to trace files affects performance. However, if a problem only occurs during production, use dynamic tracing to enable and disable tracing during the problem period to capture diagnostics for identifying the root cause.

See Also

- ["Configuring Oracle Data Provider for .NET"](#) for further details
- [OracleConfiguration Diagnostics and Tracing Properties](#)
- [settings section](#)

Database Application Migration: SQL Translation Framework

A key part of migrating non-Oracle database applications to an Oracle Database requires converting non-Oracle SQL statements to SQL statements that can be processed by an Oracle Database. SQL conversion is generally a manual and laborious process. To minimize the effort, Oracle Database 12c introduces SQL Translation Framework which takes non-Oracle SQL statements from client applications and then translates them at run-time for the Oracle Database to execute.

The SQL Translation Framework can be used to map non-Oracle stored procedure to Oracle stored procedures to ensure successful execution of those stored procedures when migrating to Oracle Database.

Currently, SQL Translation Framework is available for Sybase Adaptive Server Enterprise and Microsoft SQL Server. There is limited support for IBM DB2.

Note

SQL Translation Framework is only supported by ODP.NET, Unmanaged Driver. ODP.NET, Managed Driver and ODP.NET Core do not support this feature.

The SQL Translation Profile

The SQL Translation Profile is a database object that contains the set of captured non-Oracle SQL statements, and their translations or translation errors. The SQL Translation Profile is used to review, approve, and modify translations. A profile is associated to a single translator. However, a translator can be used in one or more SQL Translation Profiles. Typically, there is

one SQL Translation Profile per application, otherwise applications can share translated queries. You can export profiles among various databases.

1. Configuring the SQL Translation Profile Name

The default translation profile name for SQL Translation Framework can be configured through the app/web/machine .NET configuration file. If configured, connections, by default will automatically be set to the specified profile when the connection is initially created.

2. Changing the SQL Translation Profile Name

ODP.NET supports setting the profile name through the .NET config file, logon trigger, or database service. ODP.NET does not support using ALTER SESSION from an application to set the profile name.

3. Forcing Translation

Applications are strictly prohibited to execute the following SQL which forces translation of all SQL's on the database:

```
ALTER SESSION SET events = '10601 trace name context forever, level 32'
```

4. Connection Related Error Mapping

Connection Related Error Mapping can be configured through the .NET configuration file. Please note that this error mapping strictly applies to errors which could be thrown before the connection is successfully established. Once the database connection is established successfully, then these error mapping will be completely ignored and further error translation will be provided through the error mapping configured in the database.

The rules to choose an error mapping section in the configuration file are as follows:

- a. ODP.NET uses the error mapping section which matches the configured userId, dataSource, and profile, where userId and dataSource matches the corresponding values in the connection string and profile matches the defaultProfile configuration setting.
- b. If no error mapping section is found from 4.a.), then ODP.NET uses the error mapping section which matches the userId, dataSource, and profile similar to 4.a.), but with the profile that matches with the defaultErrorMappingProfile configuration setting.
- c. If still no error mapping section is found, then ODP.NET uses the global mapping, that is, <ErrorMapping profile="*">, if configured.

5. Stored Procedure Mapping.

Application must map their native stored procedure names to the corresponding Oracle stored procedure names on the translation profile in the database. The following procedure can be used to setup the mapping in the database.

```
DBMS_SQL_TRANSLATOR.REGISTER_SQL_TRANSLATION(
  PROFILE_NAME  VARCHAR2  IN
  SQL_TEXT      CLOB      IN
  TRANSLATED_TEXT CLOB      IN      DEFAULT
  ENABLE        BOOLEAN   IN      DEFAULT)
```

Example of stored procedure mapping:

```
DBMS_SQL_TRANSLATOR.REGISTER_SQL_TRANSLATION('profile_name',
  'native_sp_name',
  'oracle_sp_name');
```

① See Also

Chapter 2, SQL Translation Framework Overview and Architecture, of the *Oracle Database Migration Guide* for more information.

Allowed Logon Version Client Support

ODP.NET enables applications to specify the minimum authentication protocol that is to be used for a given application or for a given `OracleConnection` object, depending on where it's configured.

ODP.NET Supports the setting of

- `SqlNetAllowedLogonVersionClient` on the `OracleConnection` instance object
- `SqlNetAllowedLogonVersionClient` on the `OracleConfiguration` static object
- `SQLNET.ALLOWED_LOGON_VERSION` and `SQLNET.ALLOWED_LOGON_VERSION_CLIENT` in the `sqlnet.ora` file.

① See Also

- Oracle Database Net Services Reference for details on the semantics of each configuration setting.
- [SqlNetAllowedLogonVersionClient](#)
- [SqlNetAllowedLogonVersionClient](#)
- [OracleAllowedLogonVersionClient Enumeration](#)

4

ADO.NET Entity Framework and LINQ to Entities

This section describes ADO.NET Entity Framework and LINQ to Entities. Entity Framework is a framework for providing object-relational mapping service on data models.

This section contains these topics:

- [Overview of Entity Framework](#)
- [Language Integrated Query and Entity SQL](#)
- [Mapping Oracle Data Types to EDM Types](#)
- [Oracle Number Default Data Type Mapping and Customization](#)
- [Migrating Existing Entity Framework 5 Applications to Entity Framework 6](#)
- [Code First](#)
- [Unsupported Entity Framework Features](#)

Note

ADO.NET Entity Framework and LINQ to Entities is not supported by ODP.NET Core.

Overview of Entity Framework

ODP.NET includes support for the ADO.NET Entity Framework and LINQ to Entities. ODP.NET also supports Entity SQL.

Entity Framework is a framework for providing object-relational mapping service on data models. Entity Framework addresses the impedance mismatch between the relational database format and the client's preferred object format.

Entity Framework and LINQ provides productivity benefits for the .NET developer. It abstracts the database's data model from the application's data model. Working with object-relational data becomes easier with Entity Framework's tools. Oracle's integration with Entity Framework and LINQ enables Oracle .NET developers to take advantage of all these productivity benefits.

Note

Binding scalar parameters is supported with ODP.NET and Entity Framework. In Entity Framework, parameter binding by name is supported. Binding by position is not supported.

Entity data models can be generated from Oracle database schemas. Schemas can be generated from entity data models. These Oracle entity data models can be queried and

manipulated using Visual Studio and ODP.NET. Oracle supports Code First, Database First, and Model First modeling approaches. Specifying filters on the Visual Studio Server Explorer data connection enables the Entity Data Model Wizard to also filter Oracle database objects that are fetched and displayed.

LINQ to Entities can perform queries on the Oracle Database using ODP.NET, including using LINQ to Entities built-in functions. INSERTs, UPDATEs, and DELETEs can be executed using Oracle stored procedures, or by using the `ObjectContext.SaveChanges` method.

ODP.NET supports function import of Oracle stored procedures that Entity Framework can then execute. These Oracle function imports can return a collection of scalar, complex, and entity types, including returning an Oracle implicit result set as an entity type. Implicit result set binding is supported using Oracle `REF CURSOR`.

① See Also

- [Implicit REF CURSOR Binding](#).
- For a tutorial on how to use Entity Framework, Language Integrated Query (LINQ), and generate Data Definition Language (DDL) scripts using Model First, refer to: [Entity Framework, LINQ and Model-First for the Oracle Database](#)

Language Integrated Query and Entity SQL

Language Integrated Query (LINQ) defines a set of operators that can be used to query, project, and filter data in arrays, enumerable classes, XML, relational databases, and other data sources. One form of LINQ, LINQ to Entities, allows querying of Entity Framework data sources. ODP.NET supports Entity Framework such that the Oracle database can participate in object-relational modeling and LINQ to Entities queries.

Entity SQL is a language that enables querying of Entity Framework conceptual models. It allows querying Entity Framework entities and relationships in a format that is similar to SQL. ODP.NET supports querying Oracle databases through Entity SQL.

LINQ and Entity SQL syntax are generally data source neutral.

Mapping Oracle Data Types to EDM Types

The ODP.NET manifest file describes the primitive types, such as `VARCHAR2` and `Number`, and the Entity Data Model (EDM) types, such as `string` and `Int32`, that they map to. It also includes the facets for each EDM type.

ODP.NET does not support Time literals and canonical functions related to the Time type.

Oracle considers both `NULL` and empty strings to be `NULL` strings and are considered to be equal. Operations, such as `Equals()`, `Length()`, and `Trim()` on such strings will result in a `NULL` string.

[Table 4-1](#) maps the Oracle data types to their corresponding EDM types. The table also includes details about provider type attributes and the EDM type facets associated with each Oracle data type.

Table 4-1 Mapping of Oracle Data Types and EDM Types

Oracle Data Types	EDM Types (Primitive-TypeKind)	Provider Type Attributes: Name and Value	EDM Type Facets
Bfile	Binary	• Equal Comparable: False • Order Comparable: False	EDM Type Facets for Bfile
Binary_Double	Double	• Equal Comparable: True • Order Comparable: True	Not Applicable
Binary_Float	Single	• Equal Comparable: True • Order Comparable: True	Not Applicable
Binary_Integer	Int32	• Equal Comparable: True • Order Comparable: True	Not Applicable
Blob	Binary	• Equal Comparable: False • Order Comparable: False	EDM Type Facets for Blob
Boolean	Boolean	• Equal Comparable: True • Order Comparable: True	Not Applicable
Char	String	• Equal Comparable: True • Order Comparable: True	EDM Type Facets for Char
Clob	String	• Equal Comparable: False • Order Comparable: False	EDM Type Facets for Clob
Date	DateTime	• Equal Comparable: True • Order Comparable: True	EDM Type Facets for Date
Float	Decimal	• Equal Comparable: True • Order Comparable: True	EDM Type Facets for Float
Int	Int32	• Equal Comparable: True • Order Comparable: True	Not Applicable

Table 4-1 (Cont.) Mapping of Oracle Data Types and EDM Types

Oracle Data Types	EDM Types (Primitive-TypeKind)	Provider Type Attributes: Name and Value	EDM Type Facets
Interval Day To Second	Decimal	- Equal Comparable: True - Order Comparable: True	EDM Type Facets for Interval Day To Second
Interval Year To Month	Decimal	- Equal Comparable: True - Order Comparable: True	EDM Type Facets for Interval Year To Month
Long	String	- Equal Comparable: False - Order Comparable: False	EDM Type Facets for Long
Json	String	- Equal Comparable: True - Order Comparable: True	EDM Types Facets for JSON
Long Raw	Binary	- Equal Comparable: False - Order Comparable: False	EDM Type Facets for Long Raw
NChar	String	- Equal Comparable: True - Order Comparable: True	EDM Type Facets for NChar
NClob	String	- Equal Comparable: False - Order Comparable: False	EDM Type Facets for NClob
Nested Table		Not Applicable	Not Applicable and Not Supported
Number(1,0) Number(2,0) Number(3,0) Number(4,0) Number(5,0)	Int16	- Equal Comparable: True - Order Comparable: True	Not Applicable
Number(6,0) Number(7,0) Number(8,0) Number(9,0) Number(10,0)	Int32	- Equal Comparable: True - Order Comparable: True	Not Applicable

Table 4-1 (Cont.) Mapping of Oracle Data Types and EDM Types

Oracle Data Types	EDM Types (Primitive-TypeKind)	Provider Type Attributes: Name and Value	EDM Type Facets
Number(11,0) Number(12,0) Number(13,0) Number(14,0) Number(15,0) Number(16,0) Number(17,0) Number(18,0) Number(19,0)	Int64	- Equal Comparable: True - Order Comparable: True	Not Applicable
Number (all other cases)	Decimal	- Equal Comparable: True - Order Comparable: True	EDM Type Facets for Number
NVarchar2	String	- Equal Comparable: True - Order Comparable: True	EDM Type Facets for NVarchar2
Object		Not Applicable	Not Applicable and Not Supported
Raw	Binary	- Equal Comparable: True - Order Comparable: True	EDM Type Facets for Raw
Raw(16)	Guid	- Equal Comparable: True - Order Comparable: True	Not Applicable
Ref		Not Applicable	Not Applicable and Not Supported
ROWID	String	- Equal Comparable: True - Order Comparable: True	EDM Type Facets for ROWID
Smallint	Int16	- Equal Comparable: True - Order Comparable: True	Not Applicable
Timestamp	DateTime	- Equal Comparable: True - Order Comparable: True	EDM Type Facets for Timestamp
Timestamp with Local Time Zone	DateTime	- Equal Comparable: True - Order Comparable: True	EDM Type Facets for Timestamp with Local Time Zone

Table 4-1 (Cont.) Mapping of Oracle Data Types and EDM Types

Oracle Data Types	EDM Types (Primitive-TypeKind)	Provider Type Attributes: Name and Value	EDM Type Facets
Timestamp with Time Zone	DateTimeOffset	- Equal Comparable: True - Order Comparable: True	EDM Type Facets for Timestamp with Time Zone
UROWID (size)	Binary	- Equal Comparable: True - Order Comparable: True	EDM Type Facets for UROWID
Varchar2	String	- Equal Comparable: True - Order Comparable: True	EDM Type Facets for Varchar2
VArray		Not Applicable	Not Applicable and Not Supported
XMLType	String	- Equal Comparable: False - Order Comparable: False	EDM Type Facets for XMLType

EDM Type Facets

The following sections enumerate the EDM type facets for the preceding Oracle data types. The first column of each table displays the EDM type facet names for the Oracle data type. Subsequent columns list the facet attribute names and displays their respective values.

EDM Type Facets for Bfile

Table 4-2 EDM Type Facets for Bfile

Facet name	Attributes Name and Value
MaxLength	DefaultValue: 2147483648 Constant: True
FixedLength	DefaultValue: False Constant: True

EDM Type Facets for Blob

Table 4-3 EDM Type Facets for Blob

Facet name	Attributes Name and Value
MaxLength	DefaultValue: 2147483648 Constant: True
FixedLength	DefaultValue: False Constant: True

EDM Type Facets for Char**Table 4-4 EDM Type Facets for Char**

Facet Name	Attributes Name and Value
MaxLength	Minimum: 1 Maximum: 2000 DefaultValue: 2000 Constant: False
Unicode	DefaultValue: False Constant: True
FixedLength	DefaultValue: True Constant: True

EDM Type Facets for Clob**Table 4-5 EDM Type Facets for Clob**

Facet Name	Attributes Name and Value
MaxLength	DefaultValue: 2147483647 Constant: True
Unicode	DefaultValue: False Constant: True
FixedLength	DefaultValue: False Constant: True

EDM Type Facets for Date**Table 4-6 EDM Type Facets for Date**

Facet Name	Attributes Name and Value
Precision	Constant: True DefaultValue: 0

EDM Type Facets for Float**Table 4-7 EDM Type Facets for Float**

Facet name	Attributes Name and Value
Precision	Minimum: 0 Maximum: 126 DefaultValue: 0 Constant: False

Table 4-7 (Cont.) EDM Type Facets for Float

Facet name	Attributes Name and Value
Scale	Minimum: 0
	Maximum: 38
	DefaultValue: 0
	Constant: False

EDM Type Facets for Interval Day To Second**Table 4-8 EDM Type Facets for Interval Day To Second**

Facet name	Attributes Name and Value
Precision	Minimum: 1
	Maximum: 251
	DefaultValue: 251
	Constant: False
Scale	Minimum: 0
	Maximum: 9
	DefaultValue: 0
	Constant: False

Note

EDM types do not support TimeSpan.

Use Decimal to represent the total number of seconds. An application can obtain a TimeSpan by using the TimeSpan.FromSeconds static method.

EDM Type Facets for Interval Year To Month**Table 4-9 EDM Type Facets for Interval Year To Month**

Facet name	Attributes Name and Value
Precision	Minimum: 1
	Maximum: 250
	DefaultValue: 250
	Constant: False
Scale	Minimum: 0
	Maximum: 9
	DefaultValue: 0
	Constant: False

EDM Type Facets for JSON**Table 4-10 EDM Type Facets for Json**

Facet name	Attributes Name and Value
MaxLength	DefaultValue: 33,554,432 (32M) Constant: True
Unicode	DefaultValue: True Constant: True
FixedLength	DefaultValue: False Constant: True

EDM Type Facets for Long**Table 4-11 EDM Type Facets for Long**

Facet name	Attributes Name and Value
MaxLength	DefaultValue: 2147483647 Constant: True
Unicode	DefaultValue: False Constant: True
FixedLength	DefaultValue: False Constant: True

EDM Type Facets for Long Raw**Table 4-12 EDM Type Facets for Long Raw**

Facet name	Attributes Name and Value
MaxLength	DefaultValue: 2147483647 Constant: True
FixedLength	DefaultValue: False Constant: True

EDM Type Facets for NChar**Table 4-13 EDM Type Facets for NChar**

Facet name	Attributes Name and Value
MaxLength	Minimum: 1 Maximum: 1000 DefaultValue: 1000 Constant: False

Table 4-13 (Cont.) EDM Type Facets for NChar

Facet name	Attributes Name and Value
Unicode	DefaultValue: True Constant: True
FixedLength	DefaultValue: True Constant: True

Note

For NChar, the actual data is subject to the maximum byte limit of 2000.

The value of 1000 for Maximum and DefaultValue allows the EDM wizard to display columns of NCHAR(1000), where 1000 is the maximum number of characters allowed in DDL.

EDM Type Facets for NClob**Table 4-14 EDM Type Facets for NClob**

Facet name	Attributes Name and Value
MaxLength	DefaultValue: 2147483647 Constant: True
Unicode	DefaultValue: True Constant: True
FixedLength	DefaultValue: False Constant: True

EDM Type Facets for Number**Table 4-15 EDM Type Facets for Number**

Facet name	Attributes Name and Value
Precision	Minimum: 1 Maximum: 38 DefaultValue: 38 Constant: False
Scale	Minimum: 0 Maximum: 38 DefaultValue: 0 Constant: False

EDM Type Facets for NVarchar2

Table 4-16 EDM Type Facets for NVarchar2

Facet name	Attributes Name and Value
MaxLength	Minimum: 1 Maximum: 2000 DefaultValue: 2000 Constant: False
Unicode	DefaultValue: True Constant: True
FixedLength	DefaultValue: False Constant: True

Note

For NVARCHAR2, the actual data is subject to the maximum byte limit of 4000.

The value of 2000 for Maximum and DefaultValue allows the EDM wizard to display columns of NVARCHAR2(2000), where 2000 is the maximum number of characters allowed in DDL.

EDM Type Facets for Raw**Table 4-17 EDM Type Facets for Raw**

Facet name	Attributes Name and Value
MaxLength	Minimum: 1 Maximum: 2000 Constant: False
FixedLength	DefaultValue: False Constant: True

EDM Type Facets for ROWID**Table 4-18 EDM Type Facets for ROWID**

Facet name	Attributes Name and Value
MaxLength	DefaultValue: 18 Constant: True
Unicode	DefaultValue: False Constant: True
FixedLength	DefaultValue: True Constant: True

EDM Type Facets for Timestamp

Table 4-19 EDM Type Facets for Timestamp

Facet name	Attributes Name and Value
Precision	Minimum: 0 Maximum: 9 DefaultValue: 6 Constant: False

EDM Type Facets for Timestamp with Local Time Zone

Table 4-20 EDM Type Facets for Timestamp with Local Time Zone

Facet name	Attributes Name and Value
Precision	Minimum: 0 Maximum: 9 DefaultValue: 6 Constant: False

EDM Type Facets for Timestamp with Time Zone

Table 4-21 EDM Type Facets for Timestamp with Time Zone

Facet name	Attributes Name and Value
Precision	Minimum: 0 Maximum: 9 DefaultValue: 6 Constant: False

EDM Type Facets for UROWID

Table 4-22 EDM Type Facets for UROWID

Facet name	Attributes Name and Value
MaxLength	DefaultValue: 4000 Constant: True
FixedLength	DefaultValue: True Constant: True

EDM Type Facets for Varchar2

Table 4-23 EDM Type Facets for Varchar2

Facet name	Attributes Name and Value
MaxLength	Minimum: 1 Maximum: 4000 DefaultValue: 4000 Constant: False
Unicode	DefaultValue: False Constant: True
FixedLength	DefaultValue: False Constant: True

EDM Type Facets for XMLType**Table 4-24 EDM Type Facets for XMLType**

Facet name	Attributes Name and Value
MaxLength	DefaultValue: 2147483647 Constant: True
Unicode	DefaultValue: True Constant: True
FixedLength	DefaultValue: False Constant: True

Oracle Number Default Data Type Mapping and Customization

This section describes the default number mapping behavior and how to customize it for your application. You can configure a custom mapping in the .NET configuration file to override the default mapping for each Oracle NUMBER(*p*, *0*), which represents integer values.

Oracle NUMBER data types that represent integers do not have a matching .NET integer data type with exactly the same range of acceptable values. ODP.NET uses a default mapping that ensures any .NET integer type values can be stored within the Oracle database without requiring custom data type mapping. However, it is possible that Oracle NUMBER(*p*, *0*) column data can be larger than what a .NET data type can hold when retrieving values from the database.

For example, in Entity Framework 6, Oracle NUMBER(3, 0) has a default mapping to .NET Byte. Oracle NUMBER(3, 0) can store a value up to 999, while a .NET BYTE can store up to the value of 255. If you expect the Oracle data to exceed 255, modify the mapping to a larger numeric data type, such as a .NET Int16. Setting up this custom mapping allows you to consume the data in .NET without encountering an error. When such a custom mapping is used, be cautious not to insert a .NET Int16 value beyond what an Oracle NUMBER(3, 0) column can hold. Trying to insert Int16.MaxValue (i.e. 32,767) into a NUMBER(3, 0) column will cause an Oracle Database error.

Entity Framework 6 Mapping and Customization

Managed and unmanaged ODP.NET Entity Framework 6 applications can set up custom data type mapping using a .NET configuration file. The mapping works with Code First, Database First, and Model First use cases. This format improves on the older version as it unifies how managed and unmanaged ODP.NET set their configuration values and supports auto-completion.

The following is an example of an `edmMappings` section for ODP.NET, Managed Driver:

```
<oracle.manageddataaccess.client>
  <version number="*">
    <edmMappings>
      <edmNumberMapping>
        <add NETType="bool" MinPrecision="1" MaxPrecision="1" DBType="Number" />
        <add NETType="byte" MinPrecision="2" MaxPrecision="3" DBType="Number" />
        <add NETType="int16" MinPrecision="4" MaxPrecision="5" DBType="Number" />
        <add NETType="int32" MinPrecision="6" MaxPrecision="10" DBType="Number" />
        <add NETType="int64" MinPrecision="11" MaxPrecision="19" DBType="Number" />
      </edmNumberMapping>
    </edmMappings>
  </version>
</oracle.manageddataaccess.client>
```

Where:

- DBType is the Oracle Database data type
- NETType is the .NET data type that the Oracle data type maps to
- MinPrecision is the minimum range the Oracle data type will map to the .NET type
- MaxPrecision is the maximum range the Oracle data type will map to the .NET type

The following is an example of an `edmmappings` section for ODP.NET, Unmanaged Driver. It is exactly same format as the managed driver with the exception of the opening and closing tags.

```
<oracle.unmanageddataaccess.client>
  <version number="*">
    <edmMappings>
      <edmNumberMapping>
        <add NETType="bool" MinPrecision="1" MaxPrecision="1" DBType="Number" />
        <add NETType="byte" MinPrecision="2" MaxPrecision="3" DBType="Number" />
        <add NETType="int16" MinPrecision="4" MaxPrecision="5" DBType="Number" />
        <add NETType="int32" MinPrecision="6" MaxPrecision="10" DBType="Number" />
        <add NETType="int64" MinPrecision="11" MaxPrecision="19" DBType="Number" />
      </edmNumberMapping>
    </edmMappings>
  </version>
</oracle.unmanageddataaccess.client>
```

New Default Mappings

For Entity Framework 6, ODP.NET 12.1.0.2 introduces new default mappings that apply to Code First, Database First, and Model First scenarios. These changes were necessary to support Code First interoperability.

- .NET Booleans map to Oracle Number (1,0) and vice-versa by default
- .NET Bytes map to Oracle Number (2,0) and Number (3,0) and vice-versa by default

This default behavior can be changed by providing an alternative data type mapping by configuring the section of the .NET config file.

Starting with ODP.NET version 23 and Oracle AI Database 26ai, Oracle Boolean columns map to .NET Booleans in Database First scenarios while .NET Booleans remain mapped to Oracle Number(1) in Code First scenarios.

To override the default Code First mapping, specify the Entity Framework Fluent API `HasColumnType("boolean")` for the entity property.

Data Type Mapping and Customization Process

To enable custom mapping, add the mapping information to the .NET config file *prior* to EDM creation.

If the EDM was created already before providing the mapping information, then you can *modify* the mappings either through the Visual Studio tools or manually. Using Visual Studio, go to the EDM Model Browser page. Right-click on the table(s) requiring new data type mapping and select **Table Mapping** from the pop-up menu. The **Mapping Details** window will appear usually at the bottom of your screen. Update **Column Mappings** as desired.

If you need to *add* or *delete* mappings, find the **Type** values in the CSDL mapping section of your project's existing EDMX file. Add or delete those **Type** values to the .NET data types you want the application to use. In the example below, the property name types for B00LC0L and BYTEC0L are added to the CSDL and mapped to Boolean and Byte, respectively.

Example Mapping Before CSDL Customization:

```
<Property Name="INT16C0L" Type="Int16" Nullable="false" />
```

Example Mapping After CSDL Customization:

```
<Property Name="B00LC0L" Type="Boolean" Nullable="false" />
<Property Name="BYTEC0L" Type="Byte" Nullable="false" />
<Property Name="INT16C0L" Type="Int16" Nullable="false" />
```

You can employ combinations of these customization possibilities depending on your planned mapping changes. If *many* tables and *many* columns require mapping changes, it is most efficient to delete the EDMX file and regenerate the data model. If a *few* tables and *many* columns require changes, then delete the affected tables, save the EDMX file, and select **Update Model from Database...** to include those tables again. If only a *single* table and *one or two* columns require changes, then modify the EDMX either manually or by using the **Mapping Details** window.

Note

When using the EDM wizard to create a complex type from a function import, any custom EDM type mappings specified will not be applied automatically. The EDM wizard uses the default type mappings. Developers must then manually edit the resulting complex type. Developers begin this process after the complex type is generated. Any type declaration (field, property, constructor parameter, etc.) in the complex object which has an undesired type mapping, such as Decimal rather than Boolean, should be manually edited to the desired type.

StoreGeneratedPattern Enumeration

The following sections describe the Identity attribute and the Virtual column.

Identity Attribute

Oracle Database 12c (12.1) and later versions support table or view Identity attribute columns. Oracle has three Identity attribute types. When the EDM wizard generates a data model from an Oracle Identity attribute-containing table or view, ODP.NET will set the value of StoreGeneratedPattern to Identity in the .edmx file for any of three Oracle Identity types. The Identity attribute-associated column will use the server-generated value during INSERT; hence, application developers no longer need to create a sequence nor trigger. If the .NET application attempts to set the Identity attribute itself, this value will be ignored.

Virtual Column

Oracle Database versions can store expressions directly in base tables as Virtual columns, also known as Generated columns. Virtual columns cannot be inserted into or updated. ODP.NET will not automatically set StoreGeneratedPattern to Computed in the EF model for Virtual columns. To avoid errors, application developers need to add or change the value of StoreGeneratedPattern to Computed for Virtual columns after the model generation. Once done, Virtual columns are excluded from INSERTs and UPDATES upon calling SaveChanges().

Resolving Compilation Errors When Using Custom Mapping

If the custom mapping in a .NET configuration file has changed, then regenerate the data model to solve compilation errors introduced by the changes.

Under certain scenarios, custom mapping may cause compilation errors when a project that uses custom mapping is loaded by Visual Studio. One specific scenario is when Visual Studio opens a project with an existing custom mapping that now generates errors when those errors did not exist before. You may use the following workaround for such scenarios:

1. Open Visual Studio Help, About Microsoft Visual Studio. Click **OK** to exit the dialog box.
Alternatively, open the to-be-used connection in Server Explorer.
2. Compile the project again to eliminate the compilation errors.

Mapping Boolean and Guid Parameters in Custom INSERT, UPDATE, and DELETE Stored Procedures

When using your custom INSERT, UPDATE, or DELETE stored procedure in Stored Procedure Mapping, the following error might occur:

Error 2042: Parameter Mapping specified is not valid.

This can happen if a Number parameter has been mapped to a Boolean attribute, or if a RAW parameter has been mapped to a Guid attribute.

The solution is to manually add Precision="1" for the Number parameter, and MaxLength="16" for the RAW parameter of your stored procedure in the SSDL.

Migrating Existing Entity Framework 5 Applications to Entity Framework 6

To migrate existing Database First Entity Framework 5 applications to Entity Framework 6, use the following instructions. The first four steps are generic to all Entity Framework applications. The last four steps are specific to Oracle deployments.

1. Uninstall Entity Framework 5 in Visual Studio Package Manager Console. For example,
`Uninstall-Package EntityFramework`

2. Install Entity Framework 6 in Package Manager Console. For example,
`Install-Package EntityFramework -Version 6.0.2`

This step adds Entity Framework 6 to the `configSections` entry and adds a new section called `entityFramework`.

3. Delete the following namespaces from your application:

```
// C#  
using System.Data.EntityClient;  
using System.Data.Objects;
```

4. Add the following namespaces to your application:

```
// C#  
using System.Data.Entity.Core.EntityClient;  
using System.Data.Entity.Core.Objects;
```

5. Add the Oracle Entity Framework 6 provider configuration information to the .NET config file in the providers section. Modify the ODP.NET version if using a version besides 6.121.2.0. If you installed the ODP.NET NuGet package, you can skip this step as the NuGet install has already added made this change.

```
<provider invariantName="Oracle.DataAccess.Client"  
type="Oracle.DataAccess.EntityFramework.EFOracleProviderServices,Oracle.DataAccess.En  
tityFramework, Version=6.121.2.0, Culture=neutral,  
PublicKeyToken=89b483f429c47342" />
```

```
<provider invariantName="Oracle.ManagedDataAccess.Client"  
type="Oracle.ManagedDataAccess.EntityFramework.EFOracleProviderServices,Oracle.Manage  
dDataAccess.EntityFramework, Version=6.121.2.0, Culture=neutral,  
PublicKeyToken=89b483f429c47342" />
```

6. Add the `Oracle.ManagedDataAccess.EntityFramework` or `Oracle.DataAccess.EntityFramework` assembly as a reference to the project.
7. Modify the Oracle data type to .NET data type mappings as required by your application. See "[Entity Framework 6 Mapping and Customization](#)" for more details.
8. Rebuild the application.

Code First

Using the Entity Framework Code First modeling path, developers define the application domain model using source code rather than working directly with a designer or an XML-based configuration file. The classes defined within the source code become the model. The Code First model path offers an alternative to the existing Entity Framework Database First and Model First paths. Within Code First, the classes defined in code that comprise the model are

known as Plain Old CLR Objects (POCOs). This name derives from the fact that these classes have no dependency upon Entity Framework itself and are independent of it.

Oracle's support for the Code First modeling path enables .NET developers to take advantage of Oracle Database benefits.

① See Also

[Configuring for Entity Framework Code First](#)

Mapping of .NET Types to Oracle Types

When using the Code First path, the model is defined by the application's classes and properties. The property data types need to be mapped to the Oracle Database table data types. The following table lists the default mapping of supported .NET types to Oracle types as well as how to map a String property to non-default Oracle types:

Table 4-25 Mapping of .NET Data Types to Oracle Data Types

.NET Data Type	Oracle Data Type	Mapping Method
Boolean	number(1, 0)	Default Note: The default mapping of Boolean/Byte types may be specified in the EDM Mapping configuration. Reference the EDM Mapping sections in the documentation for additional information.
Byte	number(3, 0)	Default Note: The default mapping of Boolean/Byte types may be specified in the EDM Mapping configuration. Reference the EDM Mapping sections in the documentation for additional information.
Byte[]	blob	Default
Int16	number(5, 0)	Default Note: The default mapping of integer types may be specified in the EDM Mapping configuration. Reference the EDM Mapping sections in the documentation for additional information.
Int32	number(10, 0)	Default Note: The default mapping of integer types may be specified in the EDM Mapping configuration. Reference the EDM Mapping sections in the documentation for additional information.

Table 4-25 (Cont.) Mapping of .NET Data Types to Oracle Data Types

.NET Data Type	Oracle Data Type	Mapping Method
Int64	number(19, 0)	Default Note: The default mapping of integer types may be specified in the EDM Mapping configuration. Reference the EDM Mapping sections in the documentation for additional information.
Decimal	number(18, 2)	Default
Single	binary_float	Default
Double	binary_double	Default
Guid	raw(16)	Default
DateTime	date	Default
DateTimeOffset	timestamp with time zone	Default
String	nclob	Default
String	clob	Set Unicode to false using IsUnicode() fluent API
String	nvarchar2	Set Max Length to <= 2000 using HasMaxLength() fluent API or MaxLength data annotation
String	varchar2	Set Max Length to <= 4000 using HasMaxLength() fluent API or MaxLength data annotation and set Unicode to false using IsUnicode() fluent API
String	nchar	Set Max Length to <= 1000 using HasMaxLength() fluent API or MaxLength annotation and Set Column Type to NCHAR using HasColumnType() fluent API or Column data annotation
String	char	Set Max Length to <= 2000 using HasMaxLength() fluent API or MaxLength annotation and Set Column Type to CHAR using HasColumnType() fluent API or Column data annotation
String	Long	Set Column Type to LONG using HasColumnType() fluent API or Column data annotation Note: The long data type is deprecated and not recommended.
String	rowid	Set Column Type to ROWID using HasColumnType() fluent API or Column data annotation

Table 4-25 (Cont.) Mapping of .NET Data Types to Oracle Data Types

.NET Data Type	Oracle Data Type	Mapping Method
String	urowid	Set Column Type to UROWID using HasColumnType() fluent API or Column data annotation

Note

The character based columns, namely, CHAR, NCHAR, VARCHAR2, NVARCHAR2 will be created using character semantics to be able to store the specified Max Length amount of characters. However, due to the Oracle database limit, these columns can store only up to 4000 bytes. As such, these columns may not be able to store 4000 characters even if Max Length is set to 4000 characters since one character may require multiple number of bytes of storage, depending on the data and the database character set. If the character data can be longer than 4000 bytes, it may be more appropriate to use CLOB or NCLOB column.

Influencing the Oracle Data Type Characteristics

The type mappings listed in the previous table represent the mappings that occur by default or what is known as convention in Entity Framework. As illustrated with the String type, you can influence the resulting Oracle Data Type for a property as well as characteristics of that data type. There are two Entity Framework methods to influence the resulting Oracle Data Type: Data Annotations and the Code First Fluent API. Data Annotations permit you to explicitly mark a class property with one or more attributes, whereas the Code First Fluent API permits you to use code rather than attributes to achieve the same goal. For additional information regarding the use of Data Annotations and the Code First Fluent API refer to the Microsoft Entity Framework documentation.

The following table illustrates the available functionality:

Table 4-26 Mapping of Data Annotations and the Code First Fluent APIs

Data Annotation	Fluent API	Purpose	Applies To
Key	HasKey	Set a property as the Primary Key.	All Scalar Types
Required	IsRequired	Set the database column as NOT NULL.	All
MaxLength	HasMaxLength	Specifies the maximum length of the property.	String
NotMapped	Ignore	Indicates the property is not mapped to a database column.	All

Table 4-26 (Cont.) Mapping of Data Annotations and the Code First Fluent APIs

Data Annotation	Fluent API	Purpose	Applies To
ConcurrencyCheck	IsConcurrencyToken	Indicates the column should be used for optimistic concurrency checking. Note: Do not use with an unbounded (no maximum length specified) string property as this will create a LOB column. Use of a LOB column in the concurrency check will result in an ORA-00932: inconsistent datatypes error.	All
TimeStamp	IsRowVersion	Indicates to create the column as a rowversion column.	Not Supported
Column	HasColumnType	Indicates the provider-specific type to use for the database column. Note: Must be a legal compatible type. For example a Date property is not legal to map to a number column. Use the TypeName property with the Column Data Annotation to specify the type.	All
N/A	IsUnicode	Indicates to create the column as an N-type, that is, nvarchar2 or nclob. Default is true. Note: There is no Data Annotation equivalent for IsUnicode.	String
N/A	HasPrecision	Indicates the precision and scale for a decimal property. Note: There is no Data Annotation equivalent for HasPrecision.	Decimal

Code First Migrations

The Oracle Data Provider for .NET supports Code First Migrations functionality. The use of Code First Migrations with Oracle Database is supported through the Package Manager Console window migrations commands. For information on these commands, refer to the Microsoft Code First Migrations documentation:

<https://docs.microsoft.com/en-us/ef/ef6/modeling/code-first/migrations/>

Code First Migrations utilizes a table known as the Migration History table for tracking migration operations as well as model changes. ODP.NET creates this table, by default, in the user schema specified in the context connection string. This table is named

__MigrationHistory.

This table can be created in another user schema besides the user specified in the context connection string. This is accomplished through a process known as Migration History Table Customization, which is described in the following Microsoft documentation.

<https://docs.microsoft.com/en-us/ef/ef6/modeling/code-first/migrations/history-customization>

Note

- Changing the user schema for the table is the only supported customization.
- Code First Automatic Migrations is limited to working with the dbo schema only. Due to this limitation it is recommended to use code-based migrations, that is, add explicit migrations through the Add-Migration command.

Code First Migrations With No Supporting Code Migration File

When using Code First Migrations with ODP.NET, the migration history table may be dropped if no supporting code migration file existed prior to updating the database. Developers should ensure the supporting code migration file has been added prior to updating the database.

The following steps can remove the migration history table:

1. Execute application to create database objects
2. **Enable-Migrations** in the Package Manager Console
3. Make code change to POCO
4. **Update-Database** in the Package Manager Console

The following steps ensure the code migration file is created:

1. Execute application to create database objects
2. **Enable-Migrations** in the Package Manager Console
3. Make code change to POCO
4. **Add-Migration** in the Package Manager Console. This step will create the necessary code migration file.
5. **Update-Database** in the Package Manager Console

Code First Database Initialization

ODP.NET supports the following Code First Database Initializer methods:

- `CreateDatabaseIfNotExists` (default if none specified)
- `DropCreateDatabaseAlways`
- `DropCreateDatabaseIfModelChanges`

- NullDatabaseInitializer
- MigrateDatabaseToLatestVersion

Due to differences in how Oracle and SQL Server define a database, database initialization actions work on all of the Oracle objects in the model. An Oracle Database is not created or dropped, rather the objects that compose the model are considered to be the database for these operations.

Oracle Database Object Creation

In order to support the client application, ODP.NET will create and maintain the required database objects. The following are the database objects created and maintained by the provider:

- Table
- Table Column
- Primary Key
- Foreign Key
- Index
- Sequence
- Trigger

For objects which directly relate to a client application object, namely, a table which represents an application class and a table column which represents a class property, the object names used are those provided by the client. These object names must conform to the object identifier length limits for Oracle Database. For example, if a class name length exceeds the valid object identifier length in Oracle Database then the `ORA-00972: identifier is too long` exception will be raised at object creation time.

For the remaining objects, ODP.NET utilizes a name generation algorithm if the supplied name length exceeds the database identifier length limit. If the supplied name length does not exceed the database limit the name is used as-is. In all cases, the object name is created as a quoted identifier in order to preserve case and any special characters which may be part of the identifier.

In cases where the provider generates a name to comply with database identifier length limits, the name is composed of the following underscore separated elements:

- A substring of the original name (from the first character)
- A numeric suffix value calculated from the original name

The following example illustrates the results of the name generation algorithm using a simple POCO in the client application:

```
public class LongSamplePocoTestClassName
{
    [Key]
    public int Id { get; set; }

    [MaxLength(64)]
    public string Name { get; set; }
}
```

The default name for the Primary Key for the resulting table will be:

`PK_LongSamplePocoTestClassNames`

As this name contains 31 characters (single byte per character), it violates the database identifier restrictions in Oracle Database 12.1 and earlier versions. In this example, we will assume we are using a pre-12.2 database version. Newer Oracle database versions support up to 128 bytes. The rewritten Primary Key name will resemble the following value:

```
PK_LongSamplePocoTes_730795129
```

The algorithm is designed to utilize as many characters as possible from the original name such that the new name does not violate the identifier length restrictions.

Controlling Table Name and Owner

Through the use of Data Annotations or the Entity Framework Fluent API you may control the table name, as well as the table owner. For example, you may choose to explicitly set the table name to conform to your organization's naming standards or if you do not wish to, use the name Entity Framework provides. The `Table` Data Annotation is used to control both the table name and the owner. When using the Fluent API, the `.ToTable` method is used to control the table name and the owner within the `OnModelCreating` override in your class which derives from `DbContext`.

The following examples use an incomplete class definition to illustrate these actions.

Setting the table name using a Data Annotation:

```
[Table("Employee")]  
public class Employee
```

Setting the table name using the Fluent API:

```
protected override void OnModelCreating(DbModelBuilder modelBuilder)  
{  
    modelBuilder.Entity<Employee>().ToTable("Employee");  
}
```

Setting the table name and the owner using a Data Annotation:

```
[Table("Employee ", Schema="TESTUSER")]  
public class Employee
```

Setting the table name and the owner using the Fluent API:

```
protected override void OnModelCreating(DbModelBuilder modelBuilder)  
{  
    modelBuilder.Entity<Employee>().ToTable("Employee", "TESTUSER");  
}
```

Note

When using Data Annotations or the Fluent API as above to set the owner, it is required to also set the name.

Setting the Default Table Owner

Rather than set the table owner for each user table, Entity Framework 6 and higher allows you to set the default owner to be used. This is done by invoking the `HasDefaultSchema` method within the `OnModelCreating` override in your class, which derives from `DbContext`.

For example, the following code will cause all user tables to be created within the `TESTUSER` schema by default:

```
protected override void OnModelCreating(DbModelBuilder modelBuilder)
{
    modelBuilder.HasDefaultSchema("TESTUSER");
}
```

Note

The owner name is case-sensitive.

See Also

Oracle AI Database Administrator's Guide.

Using the Default Connection Factory

The default connection factory allows ODP.NET connections to be created by providing an Oracle connection string to the DbContext constructor. For example, the following entry could be used to configure the ODP.NET, Managed Driver default connection factory:

```
<defaultConnectionFactory
type="Oracle.ManagedDataAccess.EntityFramework.OracleConnectionFactory,
Oracle.ManagedDataAccess.EntityFramework,
Version=6.121.2.0,
Culture=neutral,
PublicKeyToken=89b483f429c47342" />
```

When using the default connection factory, the application supplies an Oracle connection string to the DbContext base constructor as follows:

```
public class TestContext : DbContext
{
    public TestContext()
        : base("<connection string>")
    {
    }
}
```

Where <connection string> is the ODP.NET connection string. This allows the application to connect to the database using code similar to the following:

```
using (var ctx = new TestContext())
{
    ...
}
```

For additional information please see the Microsoft documentation for the IDbConnectionFactory interface:

<https://docs.microsoft.com/en-us/dotnet/api/system.data.entity.infrastructure.idbconnectionfactory>

Unsupported Entity Framework Features

The following items are not supported by the current release of the provider:

- Mapping Code First Insert, Update, Delete operations to Stored Procedures
- TimeStamp/RowVersion properties
- Custom Configuration
- Spatial Types
- Table-valued functions
- Connection Resiliency
- Oracle synonyms

The `OracleMigrationSqlGenerator` class and its members can be overridden to allow applications to generate custom SQL. However, Oracle does not support any issues caused by customer code that overrides any of the `OracleMigrationSqlGenerator` implementation. Only issues that can be reproduced without overriding `OracleMigrationSqlGenerator` are supported.

5

Oracle Data Provider for .NET Entity Framework Core

Oracle Data Provider for .NET (ODP.NET) Entity Framework (EF) Core is a database provider that allows Entity Framework Core to be used with Oracle databases.

Entity Framework Core is a cross-platform Microsoft object-relational mapper that enables .NET developers to work with relational databases using .NET objects.

ODP.NET EF Core consists of a single 100% managed code dynamic-link library, `Oracle.EntityFrameworkCore.dll`, available via a NuGet package.

Note

ODP.NET EF Core supports spatial data types in preview. It does not support other Oracle UDT, REF, and Object Table types.

This section contains these topics:

- [Oracle Entity Framework Core 10 Features](#)
- [Oracle Entity Framework Core 9 Features](#)
- [Oracle Entity Framework Core 8 Features](#)
- [Oracle Entity Framework Core 7 Features](#)
- [Application Programming Interface](#)
- [Sample Code](#)
- [Using ODP.NET Core Classes](#)
- [Logging](#)
- [Migrations](#)
- [Scaffolding Or Reverse Engineering](#)
- [Identifier Name Length and Uniqueness](#)
- [Using Large Character or Binary Data Types](#)
- [Performance Considerations](#)
- [Artificial Intelligence Vector Data Types](#)
- [Spatial Data and NetTopologySuite](#)
- [Breaking Changes](#)

Oracle Entity Framework Core 10 Features

Oracle supports the EF Core 10 features described below. This page notes the differences from standard EF Core 10 functionality and Oracle's support.

More details about each of these features standard functionality are described on the [What's New in EF Core 10 web page](#).

Spatial Data – In Preview

The Oracle database can store spatial data that represents physical locations and object shapes. ODP.NET can operate on Oracle spatial data natively.

The NetTopologySuite spatial library maps database spatial data types in EF Core. Starting with Oracle EF Core 10, Oracle database spatial types can be accessed and mapped using new Oracle EF Core NetTopologySuite support.

This feature is currently in preview and available with the `Oracle.EntityFrameworkCore.NetTopologySuite` package on NuGet Gallery.

LeftJoin and RightJoin Operators

Oracle EF Core 10 provides full support for the new .NET 10 LINQ operators `LeftJoin` and `RightJoin`, making complex JOIN statements easier to write.

- `LeftJoin`: Simplifies writing LEFT JOIN queries.
- `RightJoin`: Simplifies RIGHT JOIN queries which retain all data from the second collection and only matched data from the first.

ExecuteUpdate for Relational JSON Columns

Previously, the `ExecuteUpdate` method did not permit updates of JSON columns. Oracle EF Core 10 supports referencing JSON columns and properties within `ExecuteUpdate` method. It enables efficient document-modeled data bulk updating in the Oracle Database.

ExecuteUpdateAsync Accepts Regular Non-Expression Lambda

The `ExecuteUpdateAsync` method can execute arbitrary database update operations. Previously, an expression tree parameter provided the database row changes, making it difficult for developers to build in dynamic changes. Manually creating expression trees is challenging and error-prone.

Oracle EF Core 10 applications can pass a lambda expression to the `ExecuteUpdateAsync` method, which then can update the records conditionally.

Redacting Inlined Constants

Earlier Oracle EF Core versions log inline SQL statement parameter values, even if sensitive data logging is disabled. To enhance security, Oracle EF Core 10 redacts inlined SQL statement parameter values by default. This prevents accidental logging of sensitive information.

Optimized Count Operation on ICollection<T>

The `Count` method is optimized in Oracle EF Core 10 to use the EXISTS SQL keyword instead of COUNT when evaluating .NET collections.

Parameter Name Simplification

Query parameter names are now simpler, removing the underscore prefix (for example, `:p_city` instead of `:_city_0`). Name uniqueness is still enforced, and reserved Oracle keywords will not be simplified to avoid conflicts.

① See Also

- [Spatial Data and NetTopologySuite](#)
- [LeftJoin and RightJoin Operator Support](#)
- [ExecuteUpdate for Relational JSON Columns](#)
- [ExecuteUpdateAsync Accepts Regular Non-Expression Lambda](#)
- [Redacting Inlined Constants](#)
- [Optimize Count Operation Use on ICollection<T>](#)
- [Simplify Parameter Names](#)

Oracle Entity Framework Core 9 Features

Oracle supports the EF Core 9 features described on this page. This page notes the differences from standard EF Core 9 functionality and Oracle's support.

More details about each of these features standard functionality are described on the [What's New in EF Core 9 web page](#).

Vector Similarity Search

Oracle EF Core 9 supports `VectorDistance`, `VectorEmbedding`, and `ToVector` EF functions with dense Oracle Database vectors. Default vector type mappings change from `.NET` string to primitive array types.

Starting with Oracle EF Core 23.8, Oracle Database sparse vectors are supported as well. This release also supports a native `OracleVector` provider type for dense vectors and `OracleSparseVector` provider type for sparse vectors for use with EF Core.

Complex types: GroupBy and ExecuteUpdate

Complex types gain new Oracle EF Core support in two ways: grouping by a complex type instance and updating by using `ExecuteUpdate`.

Prune Unneeded SQL Elements

In earlier EF Core releases, generated SQL contained unnecessary elements. Oracle EF Core 9 prunes these unnecessary tables and projection columns. This results in more compact and sometimes more efficient SQL.

In table pruning, unneeded tables in SQL JOINS are removed.

In projection pruning, unneeded projection columns in subqueries are eliminated.

Force or Prevent Query Parameterization

Oracle EF Core can use the `EF.Constant` or `EF.Parameter` methods to direct the generated query to use a parameter value (`EF.Constant`) or parameter variable (`EF.Parameter`). Oracle EF Core 9 introduces `EF.Parameter` support.

Inlined Uncorrelated Subqueries

In EF Core 8, an `IQueryable` interface referenced in another LINQ is executed in a separate database round trip. This results in multiple round trips and slows performance.

Oracle EF Core 9 enables the sub-query (`IQueryable`) and the LINQ executions in a single round trip.

Queries Using Count != 0 Optimization

In Oracle EF Core 8, LINQ queries could be translated to use the SQL `COUNT` function. In EF Core 9, these LINQ queries will use the more efficient SQL `EXISTS`.

New ToHashSetAsync<T> Methods

Oracle EF Core adds an asynchronous version of `Enumerable.ToHashSet<T>`. This method returns query results as a `HashSet`.

Improved Data Seeding

Oracle EF Core has an easy way to populate a database with an initial data set. This is known as data seeding. `DbContextOptionsBuilder` introduces new `UseSeeding` and `UseAsyncSeeding` methods that are executed when the `DbContext` initializes during an `EnsureCreated` or `EnsureCreatedAsync` operation.

Auto-Compiled Models and MSBuild Integration

Compiled EF Core models enhance app startup time for large models with hundreds of entity types. Previously, compiled models were generated manually using the command line. EF Core then was directed to use them with the `UseModel` method.

Oracle EF Core 9 can detect and use compiled models automatically, eliminating the need for `UseModel`. Moreover, if entity types or `DbContext` are changed, `MSBuild` automatically updates the compiled model when the model project is built.

Make Existing Model Building Conventions More Extensible

Oracle EF Core 9 makes it easier to extend some of the public model-building conventions, such as the property mapping code.

Update ApplyConfigurationsFromAssembly to Call Non-Public Constructors

The `ApplyConfigurationsFromAssembly` method can now be instantiated from a non-public constructor, as well as the public, parameterless form of this constructor has improved error messages during failed instantiations.

Not Supported Features

The Oracle EF Core 9 provider does not support the following features:

- Protection against concurrent migrations, also known as database lock during migrations
- Read-only primitive collections

① See Also

- [Database Functions \(EF.Functions\) for Vectors](#) to learn more about Vector Similarity Search.
- [Artificial Intelligence Vector Data Types](#)
- [Complex types: GroupBy and ExecuteUpdate support](#)
- [Prune unneeded elements from SQL](#)
- [Force or prevent query parameterization](#)
- [Inlined uncorrelated subqueries](#)
- [Queries using Count != 0 are optimized](#)
- [Other query improvements](#)
- [Improved data seeding](#)
- [Auto-compiled models](#)
- [MSBuild integration](#)
- [Make existing model building conventions more extensible](#)
- [Update ApplyConfigurationsFromAssembly to call non-public constructors](#)

Oracle Entity Framework Core 8 Features

Oracle supports the EF Core 8 features described on this page. This page notes the differences from standard EF Core 8 functionality and Oracle's support.

More details about each of these features standard functionality are described on the [What's New in EF Core 8.0 web page](#).

JSON Columns

Oracle database includes comprehensive JSON document and column support, including binary JSON storage and JSON Relational Duality.

EF Core provides support for JSON columns that allows the mapping and modifying of aggregates built from .NET types to JSON documents. LINQ queries can act upon the aggregates, which will translate to the necessary query constructs to perform operations on the JSON.

Oracle EF Core 8 JSON column support includes:

- JSON column mapping
- Queries into JSON columns
- JSON column data updates
- Translate element access into JSON arrays

Oracle Database 21c and higher support JSON columns. Oracle EF Core 8 will map aggregate types to NCLOB columns instead of JSON columns when connected to earlier Oracle database versions.

Math Translations

Concrete .NET types, such as double and float, recently introduced generic math interfaces. They mirrored existing functionality in .NET Math and MathF classes.

Oracle EF Core 8 translates calls to these generic math APIs in LINQ using existing Oracle SQL translations for Math and MathF. Developers can choose between math APIs, such as either `Math.Sin` or `double.Sin`.

Oracle EF Core support all math translations except for `DegreesToRadians` and `RadiansToDegrees`.

Value Objects Using Complex Types

Oracle EF Core 8 support complex types, also known as object types. They are structured to hold multiple values, such as an address.

Complex types must be configured in the model using either mapping attributes "[ComplexType]" or by calling the "ComplexProperty" API in `OnModelCreating`.

Complex types do not map to their own tables. Instead, they are saved inline to the table columns. This matches the table sharing behavior of owned types.

Model Building - Discriminator Columns Maximum Length

In Oracle EF Core 8, string discriminator columns in table-per-hierarchy inheritance mapping are now configured with a maximum length. This length is calculated as the smallest Fibonacci number covering all defined discriminator values.

Raw SQL Queries for Unmapped Types

Oracle EF Core 8 has added raw SQL queries returning any mappable CLR type, without including that type in the EF model. Queries using unmapped types are executed using `SqlQuery` or `SqlQueryRaw`.

Lazy-Loading Enhancements

Oracle EF Core 8 adds the following lazy-loading features:

- Lazy-loading for no-tracking queries
- Explicit loading from untracked entities
- Opt-out of lazy-loading for specific navigations

Access to Tracked Entities

Oracle EF Core 8 supports new APIs for applications to lookup tracked entities by their primary, alternate, or foreign key. These APIs are accessed through the `LocalView<TEntity>` of the entity type.

Not Supported Features

The following feature is not supported by the Oracle EF Core 8 provider:

- Primitive Collections

Oracle Entity Framework Core 7 Features

Oracle Entity Framework Core 7 supports .NET 6 and 7 runtimes. It supports connecting to Oracle Database 12.2 and higher.

Oracle supports the EF Core 7 features described on this page. This page notes the differences from standard EF Core 7 functionality and Oracle's support. More details about each of these features standard functionality are described on the [What's New in EF Core 7.0 web page](#).

Stored Procedure Mapping

Using Oracle PL/SQL stored procedures to perform entity insert, update, and delete operations are supported. [Table 5-2](#) lists the supported parameter data types. Oracle PL/SQL packages and functions are not currently supported.

Note

The `HasRowsAffectedReturnValue()` API is not supported.

Query Enhancements

Oracle EF Core improves the following LINQ query translations:

- GroupBy as final operator
- GroupJoin as final operator
- GroupBy entity type
- Subqueries don't reference ungrouped columns from outer query
- Read-only collections can be used for Contains
- Translations of string aggregate functions, `string.Join` and `string.Concat`
- Translation of `string.IndexOf`
- Translation of `GetType` for entity types
- Filtered Include on hidden navigations

Model Building Conventions

Oracle EF Core now supports model building that allow applications to remove or replace any model building conventions and to add new conventions.

Model Building Enhancements

New enhancements in model building are supported with the following features and limitations:

- Mapping attribute for composite keys
- `DeleteBehavior` mapping attribute
- Properties mapped to different column names
- Unidirectional many-to-many relationships
- Entity splitting

- Descending Indexes are only supported for code first, and not for scaffolding.
- `ITableIndex.IsDescending` property, which returns a Boolean indicating whether the index is descending or not, is not supported.

Custom Reverse Engineering Templates

Oracle EF Core 7 provider supports scaffolding customization of a generated EF model.

New and Improved Interceptors and Events

Enhancements to interceptors which enable interception, modification, and/or suppression of EF Core operations are supported. These new improvements include interception for:

- New entity instance creation and population
- LINQ expression tree modification before query compilation
- Handling optimistic concurrency
- Connections before checking the connection string is populated
- After the result set has been consumed, but before the result set is closed
- `DbConnection` creation
- `DbCommand` after initialization

The new events include:

- Right before an entity is tracked or changes state
- Before and after changes are detected to entities and properties

DbContext API and Behavior Enhancements

Improvements to `DbContext` and related classes are supported.

- Suppressor for uninitialized `DbSet` properties
- Distinguish cancellation from failure in logs
- New `IProperty` and `INavigation` overloads for `EntityType` methods
- `EntityType` for shared-type entity types
- `ContextInitialized` is now logged as Debug
- `IEntityTypeGraphIterator` is publicly usable

Improved Value Generation

Improvements to the automatic generation of values for key properties are supported.

- Value generation for domain-driven design guarded types
- Sequence-based key generation

For sequence-based key generation, Oracle EF Core 7 provider introduces a new fluent API, `UseSequence()`, which can configure the default values of a key property to be generated using a database sequence.

Migrations Tooling Improvements

Improvements when using the EF Core Migrations command-line tools are supported.

This includes the `UseOracle()` method which now accepts a null connection string so that the connection string can be assigned at a later time without encountering an exception. It also includes being able to detect when migration tools are running.

First-Class Windows Forms Data Binding

The new data binding experiences introduced in Windows Forms which integrates well with EF Core are supported by the Oracle EF Core 7 provider.

Not Supported Features

The following features are not supported by the Oracle EF Core 7 provider.

- Mapping to and from JSON columns
- Bulk updates
- Faster `SaveChanges`
- TPC inheritance mapping
- Performance enhancements for proxies

Application Programming Interface

ODP.NET EF Core supports standard EF Core application programming interfaces. The provider contains additional extension methods specific to the provider.

- [DatabaseFacade Class](#)
- [DbContextOptionsBuilder Class](#)
- [IQueryingEnumerable Interface](#)
- [MigrationBuilder Class](#)
- [ModelBuilder Class](#)
- [OracleSQLCompatibility Enumeration](#)
- [Database Functions \(EF.Functions\) for Vectors](#)

DatabaseFacade Class

ODP.NET EF Core contains additional extension methods and changes to method default behavior for the `DatabaseFacade` class.

- [DatabaseFacade.IsOracle](#)
- [DatabaseFacade.EnsureCreated](#)
- [DatabaseFacade.EnsureCreated\(string\[\]\)](#)
- [DatabaseFacade.EnsureDeleted](#)
- [DatabaseFacade.EnsureDeleted\(string\[\]\)](#)

DatabaseFacade.IsOracle

This method returns true if ODP.NET is the currently used database provider.


```
// C#  
public static bool IsOracle()
```

Returns a bool value.

Note

The provider is only known after the provider is set in the DbContext.

DatabaseFacade.EnsureCreated

This property ensures that the tables for the schema defined in the current context exists.

Declaration

```
// C#  
public static bool EnsureCreated()
```

Return Value

A bool

Remarks

If any of the tables in the schema exist, then no action is taken. Pre-existing tables are not checked for compatibility with the EF Core context model.

If none of the tables in the schema exist, then all the defined context model objects are created.

If the user/schema specified in the connection string does not exist, then an error is thrown and no action is taken to create the user/schema. The administrator must create the user/schema and assign the appropriate privileges prior to using this method.

The return value is true if all the objects defined in the context are created. It is false if any of the tables for the schema already exist.

Exception

NotSupportedException() is thrown when a non-existent user/schema is specified in the connection string.

Type: NotSupportedException()

Message: Required user does not exist or invalid user name/password provided

DatabaseFacade.EnsureCreated(string[])

This property ensures that the tables for the specified schemas in the string array exist.

Declaration

```
// C#  
public static bool EnsureCreated (string[] schemas)
```

Parameters

- `schemas` – List of schemas to check for the EF Core context's pre-existing tables. Schema names are case-sensitive.

Return Value

A `bool`

Remarks

If any of the tables in the string array `schema` list exists, then no action is taken. Pre-existing tables are not checked for compatibility with the EF Core context model.

If none of the tables in the string array `schema` list exist, then all the defined context model objects are created.

If the user/schema specified in the connection string does not exist, then an error is thrown and no action is taken to create the user/schema. The administrator must create the user/schema and assign the appropriate privileges prior to using this method.

If the `schemas` passed to this method does not include the user/schema specified in the connection string, then that schema is implicitly added to the array of schemas.

If the array of schemas is null or length zero, then the `DatabaseFacade.EnsureCreated()` API is called.

The return value is `true` if all the objects defined in the context are created. It is `false` if any of the tables for the schema already exist.

Exception

`NotSupportedException()` is thrown when a non-existent user/schema is specified in the connection string.

Type: `NotSupportedException()`

Message: Required user does not exist or invalid user name/password provided

Sample Code

```
using (var db = DbContext())
{
    db.Database.EnsureCreated(new string[]{"SCOTT", "HR", "EFUser"});
}
```

DatabaseFacade.EnsureDeleted

This property ensures that all the schema user's created objects are deleted.

Declaration

```
// C#
public static bool EnsureDeleted()
```

Return Value

A `bool`

Remarks

If none of the EF Core context model objects exist, no action is taken. If any of the objects exist, then all the user/schema objects are dropped, except for Oracle data dictionary objects.

Warning: The dropped objects include schema objects outside of the EF Core context model, as long as the user/schema has privileges to drop those objects.

If the schema defined in the current context does not exist, then no action is taken.

The return value is `true` if an attempt is made to drop all user created objects related to the schema in the current context. It is `false` if the schema specified in the connection string does not exist.

DatabaseFacade.EnsureDeleted(string[])

This property ensures that the user/schema objects for the specified schemas in the string array are deleted.

Declaration

```
// C#  
public static bool EnsureCreated (string[] schemas)
```

Parameters

- `schemas` – List of schemas to drop user generated objects. Schema names are case-sensitive.

Return Value

A `bool`

Remarks

If any of the objects exist, then all the user/schema objects are dropped, except for Oracle data dictionary objects. If none of the EF Core context model objects exist, no action is taken. If the schemas passed to this method does not include the user/schema specified in the connection string, then that schema is implicitly added to the array of schemas.

Warning: The dropped objects include schema objects outside of the EF Core context model, as long as the user/schema has privileges to drop those objects.

If the specified schemas do not exist, then no action is taken.

The return value is `true` if an attempt is made to drop all user created objects that the user has privilege to in the specified schemas. It is `false` if the schema specified in the connection string does not exist.

Sample Code

```
using (var db = DbContext())  
{  
    db.Database.EnsureDeleted(new string[]{"SCOTT", "HR", "EFUser"});  
}
```

DbContextOptionsBuilder Class

ODP.NET EF Core contains additional extension methods and changes to method default behavior for the `DbContextOptionsBuilder` class.

- [OracleDbContextOptionsBuilder.UseNetTopologySuite\(double tolerance\)](#)
- [DbContextOptionsBuilder.UseOracle](#)
- [UseOracle\(string connectionString\)](#)

- [UseOracle\(DbContextOptionsBuilder, Action<OracleDbContextOptionsBuilder> oracleOptionsAction = null\)](#)
- [UseOracleSQLCompatibility\(enum version\)](#)

OracleDbContextOptionsBuilder.UseNetTopologySuite(double tolerance)

This extension method enables spatial mapping using NetTopologySuite in Oracle EF Core.

Declaration

```
// C#
public static OracleDbContextOptionsBuilder UseNetTopologySuite(
    this OracleDbContextOptionsBuilder optionsBuilder, double tolerance = 0.005)
```

Parameter

- Tolerance - The minimum distance at which two points or features are considered distinct for spatial calculations.

Remarks

- Call this method during DbContext configuration.
- If a Tolerance value is not specified, then the default Tolerance value is 0.005 . The value must be greater than zero.

DbContextOptionsBuilder.UseOracle

This extension method sets the provider and database connection configuration to connect to Oracle Database. Developers can set any connection string attributes that are available in ODP.NET Core. The available method overloads that can be called are as follows:

- `UseOracle(string connectionString)`
- `UseOracle(string connectionString, Action<OracleDbContextOptionsBuilder> oracleOptionsAction = null)`
- `UseOracle(DbConnection connection, Action<OracleDbContextOptionsBuilder> oracleOptionsAction = null)`
- `DbContextOptionsBuilder<TContext> UseOracle<TContext>(string connectionString, Action<OracleDbContextOptionsBuilder> oracleOptionsAction = null)`
- `DbContextOptionsBuilder<TContext> UseOracle<TContext>(DbConnection connection, Action<OracleDbContextOptionsBuilder> oracleOptionsAction = null)`
- `UseOracle(DbContextOptionsBuilder, Action<OracleDbContextOptionsBuilder> oracleOptionsAction = null)`

UseOracle(string connectionString)

This extension method sets the provider and database connection configuration. Developers can set any connection string attributes that are available in ODP.NET Core.

Declaration

```
// C#
optionsBuilder.UseOracle(@"User Id=blog;Password=<password>;Data Source=pdborcl;");
```

UseOracle(DbContextOptionsBuilder, Action<OracleDbContextOptionsBuilder> oracleOptionsAction = null)

The following extension configures the EF Core context to connect to an Oracle database without initially setting any `DbConnection` nor connection string. The `DbConnection` or connection string must be set before the `DbContext` attempts to connect to a database. To set the connection using, use `RelationalDatabaseFacadeExtensions.SetDbConnection` or `RelationalDatabaseFacadeExtensions.SetConnectionString`.

Declaration

```
// C#  
public static DbContextOptionsBuilder UseOracle(this DbContextOptionsBuilder,  
Action<OracleDbContextOptionsBuilder>)
```

Parameters

- `DbContextOptionsBuilder` - The builder being used to configure the context
- `Action<OracleDbContextOptionsBuilder>` - An optional action to allow additional Oracle specific configuration

Return Value

The options builder so that further configuration can be chained.

Sample Code

```
// C# - Setting up the DB context  
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder) =>  
optionsBuilder.UseOracle();  
// Using the DB context  
using (var context = new DbContext())  
{  
context.Database.SetDbConnection(new OracleConnection(<connection string>));  
}
```

① Note

- `optionsBuilder` is of type `DbContextOptionsBuilder`.
- Do not use Oracle built-in accounts to store Entity Framework Migrations.

UseOracleSQLCompatibility(enum version)

This extension method specifies the database version generated SQL and functionality to be compatible with.

With Oracle EF Core 21.12.1 and higher, `UseOracleSQLCompatibility` extension method now takes an enumeration type as an argument rather than a string value.

The valid enum values that can be used are:

- `OracleSQLCompatibility.DatabaseVersion19`
- `OracleSQLCompatibility.DatabaseVersion21`

- `OracleSQLCompatibility.DatabaseVersion23`

By default, the SQL compatibility value will be `OracleSQLCompatibility.DatabaseVersion21`. So generated SQL will be compatible with database version 21. If using Oracle EF Core 8 with `OracleSQLCompatibility.DatabaseVersion21`, JSON database columns are supported. Otherwise, aggregate types will map to NCL0B columns in the database instead of JSON.

Using `OracleSQLCompatibility.DatabaseVersion23`, the .NET bool type will map to Oracle BOOLEAN column type rather than NUMBER(1) by default. To map .NET bools to NUMBER(1) by default, use `OracleSQLCompatibility.DatabaseVersion21` or lower.

The default enumeration value matches the ODP.NET version. For ODP.NET 21c, the default is `OracleSQLCompatibility.DatabaseVersion21`.

The following example shows how to set `UseOracleSQLCompatibility`.

```
// C#
optionsBuilder.UseOracle("User Id=hr;Password=<password>;Data Source = inst1", b =>
b.UseOracleSQLCompatibility(OracleSQLCompatibility.DatabaseVersion19));
```

Note

- `optionsBuilder` is of type `DbContextOptionsBuilder`.
- This extension method exists for EF Core 8 only.

IQueryingEnumerable Interface

This section includes:

- [IQueryingEnumerable.ToQueryString Extension Method](#)

IQueryingEnumerable.ToQueryString Extension Method

A string representation of the Oracle SQL query used. This extension method will generate SQL that can be run in Oracle Database and Oracle Autonomous Database.

To execute the generated SQL programmatically, developers can adapt the following C# pseudo-code for their specific requirements. The pseudo-code demonstrates how to generate the script using `ToQueryString()` on a sample LINQ query, and then how to execute the script with an `OracleCommand`, depending on the database version backing the application.

```
using System.Data;
using Microsoft.EntityFrameworkCore;
using Oracle.ManagedDataAccess.Client;

class ToQueryStringPseudoCode
{
    static void Main(string[] args)
    {
        using (ModelContext db = new ModelContext())
        {
            //sample LINQ to convert query string from
            string name = "Name";
            var query = db.Set<Instructor>().Where(c => c.Name == name);
            string sqltext = query.ToQueryString();
        }
    }
}
```

```

        //'sqltext' can be used directly with OracleCommand
        OracleConnection con = new OracleConnection("<Connection String>");
        con.Open();
        OracleCommand cmd = con.CreateCommand();
        cmd.CommandText = sqltext;
        OracleDataReader reader;
        reader = cmd.ExecuteReader();

        //verifying the result set
        while (reader.Read())
        {
            Console.WriteLine($"{reader[0]}, {reader[1]}, {reader[2]}, {reader[3]}");
        }
        con.Close();
    }
}
}

```

MigrationBuilder Class

MigrationBuilder.IsOracle Extension Method

Returns true if the MigrationBuilder object uses ODP.NET as its database provider.

Declaration

```
public static bool IsOracle(this MigrationBuilder)
```

Parameters

- MigrationBuilder object

Return Value

A bool.

Sample Code

```

var migrationBuilder = new MigrationBuilder("Oracle.EntityFrameworkCore");
bool b_oracle = migrationBuilder.IsOracle(); //returns true for ODP.NET

```

ModelBuilder Class

ODP.NET EF Core contains additional extension methods and changes to method default behavior for the ModelBuilder class.

- [ModelBuilder UseIdentityColumn\(\) and UseOracleIdentityColumn\(\)](#)

ModelBuilder UseIdentityColumn() and UseOracleIdentityColumn()

This extension method specifies whether the column is an identity column or have it associated with a sequence and a trigger to have a server generated column value, depending on the value passed to UseOracleSQLCompatibility(). By default, columns do not have this extension method enabled.

```

// C #
protected override void OnModelCreating(ModelBuilder modelBuilder)
{

```

```
modelBuilder.Entity<Blog>().Property(p => p.Id).UseIdentityColumn();  
}
```

For EF Core 3.1 and higher versions, use `UseIdentityColumn`. For EF Core 3.1 Core and lower versions, you can use `UseOracleIdentityColumn`. Starting with EF Core 5, `UseOracleIdentityColumn` is no longer available to use with Oracle EF Core. These two methods have identical functionality. For the remainder of this section, the term `UseIdentityColumn` is synonymous with `UseOracleIdentityColumn`.

OracleSQLCompatibility Enumeration

The `OracleSQLCompatibility` enum is used to specify database version the generated SQL should be compatible with. It is also used to determine the type mappings for .NET types.

There are currently 3 possible values of `SQLCompatibility` that can be set in the application using `UseOracleSQLCompatibility()` API.

- `OracleSQLCompatibility.DatabaseVersion19`
- `OracleSQLCompatibility.DatabaseVersion21`
- `OracleSQLCompatibility.DatabaseVersion23`

If using Oracle EF Core 8 with `OracleSQLCompatibility.DatabaseVersion21`, JSON database columns are supported. Otherwise, aggregate types will map to NCLOB columns in the database instead of JSON.

Using `OracleSQLCompatibility.DatabaseVersion23`, the .NET bool type will map to Oracle `BOOLEAN` column type rather than `NUMBER(1)` by default. To map .NET bools to `NUMBER(1)` by default, use `OracleSQLCompatibility.DatabaseVersion21` or lower.

The default enumeration value matches the ODP.NET version. For ODP.NET 21c, the default is `OracleSQLCompatibility.DatabaseVersion21`.

Database Functions (EF.Functions) for Vectors

EF Core providers can have custom C# methods that invoke database functions. These methods can be used in LINQ as `EF.Functions` extension methods. ODP.NET provides a number of them for vector data management. These vector methods can be used in EF Core 9 and higher with Oracle EF Core version 23.6 or higher. Vector features require connecting to Oracle AI Database 26ai or higher and setting `OracleSQLCompatibility` to be `DatabaseVersion23` or higher.

- [VectorDistance](#)
- [VectorEmbedding](#)
- [ToVector](#)

VectorDistance

This method calculates the distance between the two given vectors.

Declaration

```
// C# - 2 overloaded methods  
public static double VectorDistance(object expression1, object expression2)  
public static double VectorDistance(object expression1, object expression2, string  
metric)
```


Parameters

expression1 and expression2 represent the two vectors for which the distance is to be calculated.

metric is a string which indicates the metric used to calculate the distance.

Return Value

System.Double that is the distance between the two vectors in the metric format

Exception

- If any argument value is null, then an `ArgumentNullException` is thrown.
- If any argument value is empty string, then an `ArgumentException` is thrown.
- If any invalid argument value is passed, then an `OracleException` is thrown with the database error message.

Remarks

This method uses the Oracle `VECTOR_DISTANCE` database function.

Both vector parameters must have the same number of dimensions and vector format. This is a `VECTOR_DISTANCE` function requirement. They can be column expressions, variables, or inline strings.

The possible distance metric values are:

- Cosine (Default)
- Dot
- Euclidean
- Euclidean_Squared
- Hamming
- Jaccard
- Manhattan

If you want to use the metric parameter as a variable, then you must enclose it in the `EF.Constant()` method because the metric argument in the `VECTOR_DISTANCE` database function is not a string, but a metric expression.

Sample Code

```
var query1 = (from t in ctx.AIData
               where t.Id == 10
               select EF.Functions.VectorDistance(t.V, vector, "Cosine")).ToList();

var distanceType= "Cosine";
var query2 = (from t in ctx.AIData
               where t.Id == 10
               select EF.Functions.VectorDistance(t.V, vector,
               EF.Constant(distanceType))).ToList();
```

① See Also

Oracle SQL VECTOR_DISTANCE database function

VectorEmbedding

This method returns the input string's vector embedding using the specified model.

Declaration

```
// C#  
public static object VectorEmbedding(string modelName, object expression)
```

Parameters

modelName is the importing embedding model to generate the vector embedding from.

expression is the value from which to generate the embedding.

Return Value

System.Object that is the provided expression's embedding in a vector format.

Exception

- If any argument value is null, then an `ArgumentNullException` is thrown.
- If any argument value is empty string, then an `ArgumentException` is thrown.
- If any invalid argument value is passed, then an `OracleException` is thrown with the database error message.

Remarks

This method uses the Oracle VECTOR_EMBEDDING database function.

The modelName should be the same as what is in the database. The model needs to be loaded in the database prior to using this EF Core method.

modelName can be an inline string or variable. If it is a variable, enclose it in the `EF.Constant()` method to prevent parameterization. modelName is an expression, not a string in the VECTOR_EMBEDDING database function.

The expression argument can be a column, variable, or inline string.

Sample Code

```
string keyword = "cars";  
var query1 = (from t in ctx.AIData  
              select EF.Functions.VectorEmbedding("doc_model",  
keyword)).Take(1).ToList();  
  
var modelName = "doc_model"  
var query2 = (from t in ctx.AIData  
              select EF.Functions.VectorEmbedding(EF.Constant(modelName),  
keyword)).Take(1).ToList();
```

① See Also

Oracle SQL VECTOR_EMBEDDING database function

ToVector

This method converts a vector from one format to another format.

Declaration

```
// C# - 3 overloaded methods
public static object ToVector(object expression)
public static object ToVector(object expression, long dimensions)
public static object ToVector(object expression, long dimensions, string format)
```

Parameters

expression is the vector to be converted to a different vector format.

dimensions is a numeric value for the number of dimensions for the new vector.

format is the vector numeric format to convert the vector to.

Return Value

System.Object that is the in the specified vector numeric format.

Exception

- If any argument value is null, then an `ArgumentNullException` is thrown.
- If any argument value is empty string, then an `ArgumentException` is thrown.
- If any invalid argument value is passed, then an `OracleException` is thrown with the database error message.

Remarks

This method uses the Oracle `T0_VECTOR` database function.

expression could be a column, variable, or inline string in vector format.

The number of dimensions can alternatively be specified as zero (0). In this case, the dimension number is determined by expression.

The possible vector format values are:

- Binary
- Int8
- Float32
- Float64
- *

This is the target vector numeric format to convert the vector to. If format value is not specified or is an asterisk (*), the format used is `Float32`.

To use dimensions or format as variables, enclose them in the `EF.Constant()` method.

① See Also

Oracle SQL TO_VECTOR database function

Sample Code

This code sample demonstrates code necessary to create a blogging context of Blogs and Posts objects.

EF Core will create database schema tables mapping to these two objects. When the application is run, it will add a new blog entry to the Blogs table, then retrieve that entry back to the application.

```
// C#
using Oracle.EntityFrameworkCore;
using System.Collections.Generic;
using Microsoft.EntityFrameworkCore;

namespace OracleBlog
{
    class Program
    {
        public class BloggingContext : DbContext
        {
            public DbSet<Blog> Blogs { get; set; }
            public DbSet<Post> Posts { get; set; }
            protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
            {
                optionsBuilder.UseOracle(@"User Id=blog;Password=<password>;Data
Source=pdborcl;");
            }
        }
        public class Blog
        {
            public int BlogId { get; set; }
            public string Url { get; set; }
            public List<Post> Posts { get; set; }
        }
        public class Post
        {
            public int PostId { get; set; }
            public string Title { get; set; }
            public string Content { get; set; }
            public int BlogId { get; set; }
            public Blog Blog { get; set; }
        }
        static void Main(string[] args)
        {
            using (var db = new BloggingContext())
            {
                var blog = new Blog { Url = "https://blogs.example.com" };
                db.Blogs.Add(blog);
                db.SaveChanges();
            }
            using (var db = new BloggingContext())
            {
                var blogs = db.Blogs;
            }
        }
    }
}
```

```
    }  
  }  
}
```

Additional ODP.NET EF Core sample code is available on GitHub, including stored procedure and dependency injection examples.

<https://github.com/oracle/dotnet-db-samples/tree/master/samples/ef-core>

Using ODP.NET Core Classes

Developers can use the `OracleConfiguration` class and other ODP.NET Core classes in Entity Framework Core to access ODP.NET Core-specific functionality, such as the TNS ADMIN location or tracing settings.

The ODP.NET Core assembly will already be part of any Oracle EF Core project since it is a dependency of `Oracle.EntityFrameworkCore`. Most commonly, developers will add the ODP.NET Core namespace to the project:

```
// C#  
using Oracle.ManagedDataAccess.Client;
```

Then, add the desired `OracleConfiguration` property settings. These properties should be set prior to any EF Core code as `OracleConfiguration` settings must be made prior to opening an ODP.NET connection. The below example turns on tracing and sets a TNS ADMIN location which should contain the application's `tnsnames.ora` and `sqlnet.ora` files:

```
// C#  
static void Main(string[] args)  
{  
    OracleConfiguration.TraceFileLocation = @"D:\traces";  
    OracleConfiguration.TraceLevel = 7;  
    OracleConfiguration.TnsAdmin = @"D:\tnsadmin";  
  
    <Start Entity Framework Core code>  
}
```

Oracle EF Core applications can use all the properties and behavior available in ODP.NET Core.

Logging

Oracle EF Core integrates directly with EF Core logging mechanisms. Oracle EF Core logs are filtered using `DbLoggerCategory` class and `LogLevel` enumeration.

The following `DbLoggerCategory` properties can be used to filter the logs:

- `Database`
- `Database.Command`
- `Database.Connection`
- `DbLoggerCategory.Infrastructure`
- `Migrations`
- `Model`
- `Model.Validation`

- Query
- Scaffolding
- Update

The following `LogLevel` properties are available:

- `Debug`: Displays entry and exit traces. Also displays key Oracle EF Core activities and metadata, such as SQL executed, table and column metadata, and mappings.
- `Error`: Displays error related information, including the stack trace.
- `None`

To setup logging using `DebugLoggerProvider`, include the `Microsoft.Extensions.Logging.Debug.dll` assembly in your project. Similarly to setup logging using `ConsoleLoggerProvider`, include the `Microsoft.Extensions.Logging.Console.dll` assembly in your project. Next, add the following namespace in your project:

```
using Microsoft.Extensions.Logging;
```

Next, configure the `DbContext` to use the logger factory.

```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
{
    optionsBuilder.UseOracle(@<connection string>)
        .UseLoggerFactory(_myLoggerFactory);
}
```

Finally, set the `DbLoggerCategory` and `LogLevel` properties that you would like to be written to the log. In an `EnsureCreated` scenario, you can create the following `LoggerFactory`:

```
public static readonly ILoggerFactory _myLoggerFactory
    = LoggerFactory.Create(builder =>
    {
        builder
            .AddFilter((category, level) =>
                category == DbLoggerCategory.Database.Name &&
                // Filter the logs based on DbLoggerCategory. Comment this line above if you do not want
                // to filter logs based on DbLoggerCategory.
                level >= LogLevel.Trace
                // Filter the logs based on LogLevel. All levels greater than or equal to "Trace" will
                // be enabled. Comment this line above if you do not want to filter logs based on LogLevel.
            )
            .AddDebug();
    });
```

In a migrations scenario, you can create the following `LoggerFactory`:

```
public static readonly ILoggerFactory _myLoggerFactory
    = LoggerFactory.Create(builder =>
    {
        builder
            .AddFilter((category, level) =>
                category == DbLoggerCategory.Database.Command.Name &&
                // Filter the logs based on DbLoggerCategory. Comment this line above if you do not want
                // to filter logs based on DbLoggerCategory.
                level >= LogLevel.Trace
                // Filter the logs based on LogLevel. All levels greater than or equal to "Trace" will
            )
    });
```

```

be enabled. Comment this line above if you do not want to filter logs based on LogLevel.
)
    .AddConsole();
});

```

In Scaffolding scenario, use the `-verbose` option to generate traces.

```
Scaffold-DbContext .. -verbose
```

Migrations

By convention, ODP.NET EF Core maps an appropriate database data type based on the .NET data type and its characteristics. This table shows the default mappings. Fluent APIs/Annotations can be used to map the .NET types to any valid Oracle data type.

This table shows the default mappings. Fluent APIs and Data Annotations can be used to map the .NET types to any valid Oracle data type.

Table 5-1 ODP.NET Entity Framework Core Migrations Data Type Default Mappings

.NET Type Alias	.NET Data Type	Required Fluent API(s)*	Oracle Database Data Type
bool	System.Boolean	None	BOOLEAN or NUMBER(1)
sbyte	System.Sbyte	None	NUMBER(3)
byte	System.Byte	None	NUMBER(3)
short/int16	System.Int16	None	NUMBER(5)
ushort/uint16	System.UInt16	None	NUMBER(5)
int/int32	System.Int32	None	NUMBER(10)
uint32	System.UInt32	None	NUMBER(10)
decimal	System.Decimal	None	NUMBER(18,2)
long/int64	System.Int64	None	NUMBER(19)
uint64	System.UInt64	None	NUMBER(20)
float	System.Float	None	BINARY_FLOAT
double	System.Double	None	BINARY_DOUBLE
DateTime	System.DateTime	None	TIMESTAMP(7)
DateTimeOffset	System.DateTimeOffset	None	TIMESTAMP(7) WITH TIME ZONE
TimeSpan	System.TimeSpan	None	INTERVAL DAY(8) TO SECOND(7)
char	System.Char	None	NVARCHAR2(1)
byte[]	System.Byte[]	None	RAW(2000)
byte[]	System.Byte[]	HasMaxLength(x <= 2000)	RAW(x)
byte[]	System.Byte[]	HasMaxLength(x > 2000)	BLOB
string	System.String	None	NVARCHAR2(2000)

Table 5-1 (Cont.) ODP.NET Entity Framework Core Migrations Data Type Default Mappings

.NET Type Alias	.NET Data Type	Required Fluent API(s)*	Oracle Database Data Type
string	System.String	IsUnicode(false) && IsFixedLength(false)) && HasMaxLength(x > 4000)	CLOB
string	System.String	IsUnicode(true) && IsFixedLength(false)) && HasMaxLength(x > 2000)	NCLOB
string	System.String	IsUnicode(false) && IsFixedLength(false)) && HasMaxLength(x <= 4000)	VARCHAR2(size)
string	System.String	IsUnicode(true) && IsFixedLength(false)) && HasMaxLength(x <= 2000)	NVARCHAR2(size)
string	System.String	IsUnicode(false) && IsFixedLength(true) && HasMaxLength(x < 2000)	CHAR(size)
string	System.String	IsUnicode(true) && IsFixedLength(true) && HasMaxLength(x < 1000)	NCHAR(size)
string	System.String	HasColumnType("VECTOR(dimension, OR(dimension, type))")	VECTOR(dimension, type)
guid	System.Guid	None	RAW(16)

* Corresponding data annotations can also be used instead of the specified fluent APIs.

NCHAR and NVARCHAR2 use character length semantics. The number of characters for columns with one of these data types depend on the character set, NLS_NCHAR_CHARACTERSET. ODP.NET Entity Framework Core defaults to a 2-byte character set, which allows a maximum of 2000 characters for NCHAR and NVARCHAR2 columns. If a [MaxLength(4000)] data annotation or fluent API equivalent is used for a string entity property, ODP.NET will map the property to an NCLOB type because the specified length is greater than 2000 characters.

For example, if a NVARCHAR2(4000) column on the database NLS_NCHAR_CHARACTERSET or if the NVARCHAR2 column has a maximum 32 KB length (i.e. MAX_STRING_SIZE = EXTENDED), the string entity property can use the [Column] data annotation or a fluent API equivalent to map to an N-character data type, rather than mapping to the default NLOB. Here's an example of using such an annotation:

```
[Column("Name", TypeName = "NVARCHAR2(4000)")]
string EmployeeName
```

Applications may prefer N-character data types over NLOBs as they require no additional server roundtrips to fetch and update data.

Synonyms

EF Core migrations can generate arbitrary Oracle schema objects using the RelationalDatabaseFacadeExtensions class, such as using the ExecuteSqlRaw method to perform DDL on those objects. For example, to create a synonym for the Blogs table sample, the following code snippet can be used:

```
private static void SetupDatabase()
{
    using (var db = new BloggingContext())
    {
        db.Database.EnsureDeleted();
        if (db.Database.EnsureCreated())
        {
            //other code

            #region Synonym
            db.Database.ExecuteSqlRaw(
                " create synonym s for \"Blogs\";");
            #endregion

            db.SaveChanges();
        }
    }
}
```

Scaffolding Or Reverse Engineering

ODP.NET EF Core supports scaffolding the following tables and views:

- Relational tables and views
- Materialized views

By convention, ODP.NET EF Core maps an appropriate .NET data type based on the Oracle Database data type and its characteristics.

When scaffolding, developers may wish to change the default data type mappings between Oracle Database and .NET. When modifying .NET CLR types of the generated entity's property, remove the auto-generated HasColumnType() Fluent API in the model. For example, an Oracle NUMBER(4) is scaffolded to a .NET Byte data type by default. If the .NET CLR data type is changed to .NET Short or Int16 data type, then an invalid cast exception at runtime may occur until the HasColumnType("NUMBER(4)") has been removed.

Data Type Mapping

This table shows the default mappings.

Table 5-2 ODP.NET Entity Framework Core Reverse Engineering Data Type Default Mappings

Oracle Database Data Type	.NET Type Alias	.NET Data Type
NUMBER(1)	bool	System.Boolean
NUMBER(2) to NUMBER(4)	byte	System.Byte
NUMBER(5)	short/int16	System.Int16
NUMBER(6) to NUMBER(10)	int/int32	System.Int32
NUMBER(11) to NUMBER(19)	long/int64	System.Int64
NUMBER(>19)	decimal	System.Decimal
NUMBER(p,s)	decimal	System.Decimal
NUMBER	decimal	System.Decimal
BINARY_FLOAT	float	System.Float
BINARY_DOUBLE	double	System.Double
BOOLEAN	bool	System.Boolean
TIMESTAMP	DateTime	System.DateTime
TIMESTAMP WITH TIMEZONE	DateTimeOffset	System.DateTimeOffset
TIMESTAMP WITH LOCAL TIMEZONE	DateTime	System.DateTime
DATE	DateTime	System.DateTime
INTERVAL DAY TO SECOND	TimeSpan	System.TimeSpan
INTERVAL YEAR TO MONTH	string	System.String
VARCHAR2	string	System.String
JSON	string	System.String
NVARCHAR2	string	System.String
CHAR	string	System.String
NCHAR	string	System.String
CLOB	string	System.String
NCLOB	string	System.String
RAW	byte[]	System.Byte[]
BLOB	byte[]	System.Byte[]
XMLTYPE	string	System.String
ROWID	string	System.String
UROWID	string	System.String
LONG	string	System.String
BFILE	byte[]	System.Byte[]
LONG RAW	byte[]	System.Byte[]
VECTOR	string	System.String

Scaffolding Tables from Another Schema

Developers can scaffold tables from other schemas other than the user/schema they are connected with. The connected user requires privileges to access the other schema's objects. Once these privileges are granted, developers can use the Package Manager Console (PMC) tools for Entity Framework Core to perform the scaffolding operation. For example:

```
Scaffold-DbContext "User Id=scott;Password=<password>;Data Source=myhost:1521/mydb;"
Oracle.EntityFrameworkCore -Schemas HR -Tables EMPLOYEES
```

Developers can use the `-Schemas` and `-Tables` parameters to specify which schemas and tables/views to scaffold for an Entity Framework Core model. If connecting with one user/schema to create tables from a second schema, the user must have at least `SELECT` privileges for that second schema.

The following table displays the ODP.NET Entity Framework Core behavior when the `-Schemas` and/or `-Tables` parameter is specified or left as default while scaffolding a pre-existing model using the Package Manager Console command, `Scaffold-DbContext`. All sample command excerpts below use `Scaffold-DbContext` syntax. Similar functionality is available using the EF Core tools command, `dotnet ef dbcontext scaffold`.

Table 5-3 Schema and Table Filter Mapping

Mapping of Schema and Table Filters	No Schema Filter	Schema Filter
No Table Filter	Generates all tables/views within current user/schema <i>Box 1</i>	Generates all tables/views in specified user(s)/schema(s) <i>Box 2</i>
Table Filter	Generates specified tables/views within current user/schema <i>Box 3</i>	Generates specified tables/views within current schema <i>and</i> all tables/views in specified user(s)/schema(s) Filtered results include combined <i>Box 2</i> and <i>Box 3</i> results.

It is possible to scaffold tables/views in other schemas by appending the schema name in front: `-Tables <schema>.<table/view>`

For example, the following snippet would scaffold three tables in three different schemas:

```
-Tables SCHEMA1.TABLEA, SCHEMA2.TABLEB, SCHEMA3.TABLEC
```

If the schema or table name contains any special characters, such as a period, then use brackets to delimit the schema and table names when using the `-Tables` option.

```
-Tables [SCHEMA1].[TABLEA]
```

Scaffolding Views

Oracle supports scaffolding database relational and materialized views only. These views can be either read-only or updatable. Use the `-Tables` parameter to specify the views to scaffold. If the parameter is left blank (default), all the user tables, relational views, and materialized views will be scaffolded.

Customizing Data Type Mapping

If default data type mappings are not the best fit for the .NET application, the scaffolded data type mappings can be changed. For example, the NUMBER(1) maps to a .NET bool by default. If the column data contains numbers other than Boolean values, a better mapping would be to a .NET byte.

To customize mappings, Oracle EF Core apps should use reverse engineering templates, or T4 text templates. These templates provide an alternative mapping Oracle EF Core can use during scaffolding. A common set of alternative data type mappings are in the table below. They are a more complementary match between Oracle database types and .NET types for scaffolding.

Table 5-4 ODP.NET Entity Framework Core Scaffolding Default and Alternative Data Type Mappings

Oracle Database Data Type	Default .NET Type Alias	More Complementary .NET Type Alias
NUMBER(1)	bool	byte
NUMBER(2) to NUMBER(4)	byte	Int16
NUMBER(5)	Int16	Int32
NUMBER(6) to NUMBER(10)	Int32	Int64
NUMBER(10) to NUMBER(19)	Int64	Decimal

One advantage of the more complementary .NET data types is they can store all possible values of the mapped Oracle Database data types.

Using T4 Templates for Data Type Mapping Customization

Standard .NET T4 templates provide the data type mapping customization information for EF Core scaffolding. To use the template, follow these steps.

1. Install the EF Core templates package. For example, you can incorporate the package using the .NET CLI command:

```
dotnet new install Microsoft.EntityFrameworkCore.Templates
```

2. Add the Oracle EF Core default templates to your project. This too can be done with the .NET CLI as seen here:

```
dotnet new ef-templates
```

You should now see a new project directory Code-Templates/EFCore with two files, DbContext.t4 and EntityType.t4. The DbContext.t4 template is used to scaffold a DbContext class for the database. The EntityType.t4 template is used to scaffold entity type classes for each table and view in the database. We will use the EntityType.t4 template in the next step.

3. Using the EntityType.t4 template, any CLR Type can be mapped to a specific Oracle database type. The template snippet maps the NUMBER(1) column Col1 to a .NET short data type instead of .NET bool default type:

```
//Specify the store type custom mapping to your chosen the CLR type
var clrType = (property.Name == "Col1" && property.GetColumnType() ==
"NUMBER(1)")
? typeof(short)
```

```

        : property.ClrType;

//Add any required 'usings' for the clrType
usings.AddRange(code.GetRequiredUsings(clrType));

        var needsNullable = Options.UseNullableReferenceTypes && property.IsNullable
&& !clrType.IsValueType;
        var needsInitializer = Options.UseNullableReferenceTypes && !
property.IsNullable && !clrType.IsValueType;
        #>
        public <# code.Reference(clrType) #><# needsNullable ? "?" : "" #> <#
property.Name #> { get; set; }<# needsInitializer ? " = null!;" : "" #>
        <#
        firstProperty = false;

```

The newly added template overrides Oracle EF Core's default type mapping.

Oracle provides sample T4 text templates developers can use as is or customize on the Oracle .NET GitHub sample code site. Snippets can be copied directly into an application's T4 text template with a minimum of customization.

The property name in the template is what scaffolding generates as per the standard property naming scheme. In the above example, the database column, COL1, becomes the Col1 .NET property name. In the below example, the database column C01_SMALLINT becomes the property name C01SmallInt.

```

var clrType = (property.Name == "C01SmallInt" && property.GetColumnType() ==
"NUMBER(38)")
    ? typeof(short)
    : property.ClrType;

```

An easy way to determine the .NET type name is to perform scaffolding without a template.

If altering some property aspects, such as the CLR Type, ensure the appropriate altered variable is used when printing the property as seen in the example clrType reference, not property.clrType.

```

public <# code.Reference(clrType) #><# needsNullable ? "?" : "" #> <#
property.Name #> { get; set; }<# needsInitializer ? " = null!;" : "" #>

```

Using T4 Templates to Preserve Legacy Mapping Behavior

Default data type mappings can change based on new data types introduced in .NET or Oracle database. For example, Oracle EF Core 23ai scaffolds NUMBER(1) columns to Boolean data types and then expects only Boolean values to be used. When a non-Boolean value is used, then an error, such as "System.InvalidOperationException: No coercion operator is defined between types 'System.Nullable`1[System.Int16]' and 'System.Boolean'" occurs.

If you are upgrading from Oracle EF Core 21c to 23ai, and your database uses NUMBER(1) columns for legacy .NET numeric types like byte, short, or int, then you can use a T4 template to preserve the previous behavior and avoid unwanted HasColumnType("NUMBER(1)") fluent API generation. To do this, follow the first two steps from the [Using T4 Templates for Data Type Mapping Customization](#).

For the third step, open the CodeTemplates/EFCore/DbContext.t4 file. Find the location where propertyFluentApiCalls is initialized and replace it with the following code:

```

``t4
var storeType = property.GetColumnType();
var clrType = property.ClrType;
var isBoolOrNullableBool = clrType == typeof(bool) || clrType == typeof(bool?);
var propertyFluentApiCalls = property.GetFluentApiCalls(annotationCodeGenerator)
?.FilterChain(c =>
!(Options.UseDataAnnotations && c.IsHandledByDataAnnotations)
&& !(c.Method == "IsRequired" && Options.UseNullableReferenceTypes && !
property.ClrType.IsValueType)
&& !(c.Method == "HasColumnType" && storeType == "NUMBER(1)" && isBoolOrNullableBool));

```

Save the `DbContext.t4` file.

This T4 template change detects if a column is `NUMBER(1)`. If so, it skips adding `.HasColumnType("NUMBER(1)")` for `bool` and `bool?` data types only. No other fluent API calls are modified.

For subsequent scaffold operations, the `NUMBER(1)` column will no longer generate an error as the legacy mapping is restored. The legacy mapping can also be restored without modifying the T4 text template. The developer would need to manually remove the `.HasColumnType("NUMBER(1)")` method from the entity definition scaffolding generated. The removal has to be repeated each time scaffolding occurs.

See Also

- [Custom Reverse Engineering Templates](#)
- [Oracle EF Core sample T4 templates](#)

Identifier Name Length and Uniqueness

Oracle Database limit identifier names, such as table names, column names, and primary key names, to 128 bytes. Oracle Entity Framework Core will automatically truncate identifier names that are too long to this specified length, except for table names. This prevents both the user and Entity Framework Core from creating identifiers beyond the maximum character limit of the Oracle Database and receiving an "ORA-00972: IDENTIFIER IS TOO LONG" error in most instances.

If the identifiers use multi-byte characters, the maximum identifier length may require a setting with character expansion ratio in mind to assure that all identifiers can be created in the Oracle database. For example, if the Oracle database character set is UTF8, a single character may require up to 4 bytes. Thus, to guarantee that all identifiers can be created in an Oracle database that does not support long identifiers, the `RelationalAnnotationNames.MaxIdentifierLength` field can be set to 32 bytes (i.e. 128 bytes divided by 4). By default, Oracle EF Core `MaxIdentifierLength` has a value of 128.

Even if your EF Core model does not use identifier names longer than what your database version supports, you should consider setting `MaxIdentifierLength` to a lower value. ODP.NET and Entity Framework Core auto-generate schema object names from models. They may, on occasion, append additional characters to the names, which could exceed the maximum number of characters the database permits. By setting the `MaxIdentifierLength`, you prevent EF Core from causing the ORA-00972 error inadvertently.

Using Large Character or Binary Data Types

By default, .NET `byte[]` maps to `RAW(2000)` and .NET `string` maps to `NVARCHAR(2000)`. But if your application deals with data that larger than 2000 bytes, you can use the `Column` or the `MaxLength` data annotations or the associated fluent API to create `BLOB` and `CLOB` columns in the database, respectively.

```
// This annotation will force a BLOB column to be created.  
[Column("BLOB_COLUMN", TypeName = "BLOB")]  
public byte[] BYTE_TYPE { get; set; }
```

```
// This annotations will force a NCLOB column to be created.  
[MaxLength(65536)]  
public string NCLOB_TYPE { get; set; }
```

Without these annotations, you may hit errors, such as “ORA-01460: unimplemented or unreasonable conversion requested” or “ORA-12899: value too large for column” when modifying the data.

Performance Considerations

EF Core application performance is most optimal for binding character-based data when the .NET `String` entity property bind data type and the database column data type `Unicode` support match. If the mapping is done properly, then the application will bind the string entity property value properly as `NVARCHAR2` for a `NVARCHAR2` column or bind it as `VARCHAR2` for a `VARCHAR2` column. If the types are mismatched, then additional processing is incurred on the server side, slowing down performance.

To avoid performance related issues due to this mismatch, one of the following can be done:

- If the table already exists, use scaffolding to generate the entity classes corresponding to the relational database tables. Doing so generates the correct fluent API for each table column.
- If the entity classes exist without the corresponding database tables, use migrations to generate the corresponding database columns for the string entity properties, which will honor the `IsUnicode()` or `HasColumnType()` fluent APIs that are invoked.
- If manually creating the entity classes corresponding to the database tables, correctly map each string entity property to the `NVARCHAR2` or `VARCHAR2` column type using the appropriate `IsUnicode()` or `HasColumnType()` fluent API to avoid mismatches.

Other considerations:

- If both `IsUnicode()` and `HasColumnType()` fluent APIs are used, then the `HasColumnType()` fluent API takes precedence.
- For migrations, the .NET `String` entity property maps to `NVARCHAR2` by default. For scaffolding, both `VARCHAR2` and `NVARCHAR2` columns map to .NET `string`.
- If the database column type is `VARCHAR2`, then the `IsUnicode(false)` or `HasColumnType("VARCHAR2(<length>)")` fluent API should be used to correctly map the string entity property to the `VARCHAR2` column. This avoids the performance degradation problem.
- If a string entity property is associated with a `NVARCHAR2` column, no invocations of `IsUnicode()` nor `HasColumnType()` fluent APIs are needed. Alternatively, `IsUnicode(true)` or

`HasColumnType("NVARCHAR2(<length>)")` fluent APIs can be invoked to bind the data as `NVARCHAR2`.

- In Oracle.EntityFrameworkCore 2.19.70 and earlier versions, string entity property values were always bound as `VARCHAR2` while executing LINQ queries. The behavior changed starting with Oracle EF Core 2.19.80. The string entity property values are now bound based on the mapping specified for entity string property. An application that performed optimally with Oracle EF Core 2.19.70 can degrade in performance when upgrading to a later Oracle EF Core version.
- If you encounter a new performance problem after an Oracle EF Core upgrade, verify the string entity properties associated with `VARCHAR2` columns have not set either `IsUnicode(false)` nor `HasColumnType("VARCHAR2(<length>)")` fluent APIs nor equivalent data annotations. If so, add one of these fluent API so that the character-based data are bound using the correct type.

Artificial Intelligence Vector Data Types

Starting with version 23.8, Oracle EF Core supports developing both dense and sparse Oracle Database vectors with `VectorDistance`, `VectorEmbedding`, and `ToVector` EF functions.

ODP.NET natively supports dense vectors in .NET and EF Core with the `OracleVector` data type. Sparse vectors are supported using the `OracleSparseVector` data type. These data types are mapped by default to Oracle Database `DENSE` and `SPARSE` format vectors, respectively.

Entity Framework Core 9 or higher is required for the vector data type support.

In Oracle EF Core 23.7, only dense vectors were supported and the default vector mappings were to primitive array types that included `short[]`, `float[]`, `double[]`, and `byte[]`.

In prior releases, vector data types were mapped to .NET strings.

The following table shows the .NET common language runtime (CLR) data types that Oracle Database vectors map to by default and the possible alternative mappings.

Table 5-5 Oracle Vector EF Core Data Type Mapping

Oracle Database Data Type	Default CLR Mapping	Alternative CLR Mapping
DENSE vector (fixed format)	<code>OracleVector</code>	<code>short[]</code> , <code>float[]</code> , <code>double[]</code> , or <code>byte[]</code>
DENSE vector (flexible format)	<code>OracleVector</code>	None
SPARSE vector (all formats)	<code>OracleSparseVector</code>	None

A fixed vector format is a database vector with a specific numeric format, such as `VECTOR(100, FLOAT64)`, `VECTOR(*, FLOAT32)`, or `VECTOR(24, BINARY)`. The number of dimensions can vary.

A flexible vector format can store any numeric format, such as `VECTOR(3, *)`, `VECTOR(*, *)`, or `VECTOR(100, *, SPARSE)`. The number of dimensions can vary as well.

Migrations

When an Oracle EF Core model contains `OracleVector` or `OracleSparseVector` data types, they are mapped to Oracle Database vectors only when the `HasColumnType()` fluent API or corresponding data annotation is specified. Migrations are also allowed if entity properties are the ones defined in the alternative CLR mapping column as documented in the above table.

For sparse vectors, application must declare the complete vector column type in `HasColumnType()`, such as `HasColumnType("VECTOR(3,FLOAT64, sparse)")`.

Scaffolding

By default, Oracle Database dense vectors in any numeric format map to `OracleVector` and sparse vectors in any numeric format will map to `OracleSparseVector` for EF Core scaffolding.

Fixed dense vectors mapping can be customized to map to primitive array types, including `short[]`, `float[]`, `double[]`, and `byte[]`.

Code Sample

The following EF Core code sample demonstrates how to use ODP.NET vector types with EF Core, LINQ, and EF.Functions.

```
/*
This Oracle EF Core Code First sample shows how to create and use a database
schema with Oracle vector columns. It does the following:
1. Creates two tables, one with dense vectors and the other with sparse vectors.
It uses native ODP.NET vectors, OracleVector and OracleSparseVector.
2. Inserts two rows into each table.
3. Shows various vector EF function use cases by executing LINQ commands on the tables.
Note: Native ODP.NET vector types require the Oracle.ManagedDataAccess.Types namespace.
Requires EF Core 9 or higher.
*/

using System;
using System.Linq;
using Microsoft.EntityFrameworkCore;
using Oracle.ManagedDataAccess.Types;

namespace OracleVectorSample
{
    class Program
    {
        static void Main(string[] args)
        {
            using (var ctx = new ModelContext())
            {
                ctx.Database.EnsureDeleted();
                ctx.Database.EnsureCreated();

                short[] shortdataD = [1, 2, 3];
                float[] floatdataD = [1.1f, 2.2f, 3.3f];
                double[] doubledataD = [3.3, 2.2, 1.1];
                string stringdataD = "[3,2,1]";

                OracleVector shortVecD = new OracleVector(shortdataD);
                OracleVector floatVecD = new OracleVector(floatdataD);
                OracleVector doubleVecD = new OracleVector(doubldataD);
                OracleVector flexVecD = new OracleVector(stringdataD);

                var row1dense = new Tabdense { Id = 1, Colint8
= new OracleVector(shortdataD), Colfloat32 = new OracleVector(floatdataD),
Colfloat64 = new OracleVector(doubldataD), Colflex = new OracleVector(stringdataD) };
                var row2dense = new Tabdense { Id = 2, Colint8 = shortVecD, Colfloat32
= floatVecD, Colfloat64 = doubleVecD, Colflex = flexVecD };

                //Insert data into dense vectors.
                ctx.Tabdenses.Add(row1dense);
            }
        }
    }
}
```

```

ctx.Tabdenses.Add(row2dense);
ctx.SaveChanges();

string shortdataS = "[3, [0,1,2], [1, 2, 3]]";
string floatdataS = "[3, [0,1,2], [1.1, 2.2, 3.3]]";
string doubledataS = "[3, [0,1,2], [3.3, 2.2, 1.1]]";
string stringdataS = "[3, [0,1,2], [3, 2, 1]]";

OracleSparseVector shortVecS = new OracleSparseVector(shortdataS);
OracleSparseVector floatVecS = new OracleSparseVector(floatdataS);
OracleSparseVector doubleVecS = new OracleSparseVector(doubldataS);
OracleSparseVector flexVecS = new OracleSparseVector(stringdataS);

var row1sparse = new Tabsparse { Id = 1, Colint8
= new OracleSparseVector(shortdataS), Colfloat32 = new OracleSparseVector(floatdataS),
Colfloat64 = new OracleSparseVector(doubldataS), Colflex = new
OracleSparseVector(stringdataS) };
var row2sparse = new Tabsparse { Id = 2, Colint8 = shortVecS, Colfloat32
= floatVecS, Colfloat64 = doubleVecS, Colflex = flexVecS };

//Insert data into sparse vectors
ctx.Tabsparses.Add(row1sparse);
ctx.Tabsparses.Add(row2sparse);
ctx.SaveChanges();

//Select all table data.
var resultTabDense = ctx.Tabdenses.ToList();
var resultTabSparse = ctx.Tabsparses.ToList();

//LINQ - USING DENSE VECTORS
//Below LINQ selects the ID from TABDENSE table ordering the results based on
the vector_distance of COLINT8 column and an arbitrary "[3, 2, 1]" vector data.
/* Generated query-
  SELECT "t"."ID"
  FROM "TABDENSE" "t"
  ORDER BY VECTOR_DISTANCE("t"."COLINT8", TO_VECTOR('[3, 2, 1]', 3, INT8,
DENSE), cosine)
*/
var query1D = (from t in ctx.Tabdenses
               orderby EF.Functions.VectorDistance(t.Colint8,
EF.Functions.ToVector("[3, 2, 1]", 3, "INT8", "DENSE"), "cosine")
               select t.Id);
foreach (var row in query1D)
{
    Console.WriteLine(row);
}

short[] key1 = [3, 2, 1];
OracleVector keyV = new OracleVector(key1);

//Below LINQ is similar to above LINQ but uses the vector "[3, 2, 1]" as a
variable and
orders the results based on the vector_distance of COLFLOAT64 column and "[3, 2, 1]"
vector data.
/* Generated query-
  SELECT "t"."ID"
  FROM "TABDENSE" "t"
  ORDER BY VECTOR_DISTANCE("t"."COLFLOAT64", TO_VECTOR(:key1_1, 3, FLOAT64,
DENSE), cosine)
*/
var query2D = (from t in ctx.Tabdenses
               orderby EF.Functions.VectorDistance(t.Colfloat64,

```

```

EF.Functions.ToVector(key1, 3, "FLOAT64", "DENSE"), "cosine")
    select t.Id);
foreach (var row in query2D)
{
    Console.WriteLine(row);
}

//Below LINQ is similar to above but it passes "*" as the storage format in
which case
the output vector format of TO_VECTOR is determined by the input vector argument format.
/* Generated query-
SELECT "t"."ID"
FROM "TABDENSE" "t"
ORDER BY VECTOR_DISTANCE("t"."COLFLOAT64", TO_VECTOR(:keyV_1, 3, FLOAT64,
*), cosine)
*/
var query3D = (from t in ctx.Tabdenses
               orderby EF.Functions.VectorDistance(t.Colfloat64,
EF.Functions.ToVector(keyV, 3, "FLOAT64", "*"), "cosine")
               select t.Id);
foreach (var row in query3D)
{
    Console.WriteLine(row);
}

//Similar query as above but the vector distance is calculated between two
vector
columns (COLFLOAT32 and COLINT8) after converting the numeric format of
COLINT8 to FLOAT32 numeric format via ToVector.
/* Generated query-
SELECT "t"."ID"
FROM "TABDENSE" "t"
ORDER BY VECTOR_DISTANCE("t"."COLFLOAT32", TO_VECTOR("t"."COLINT8", 3,
FLOAT32, *), cosine)
*/
var query4D = (from t in ctx.Tabdenses
               orderby EF.Functions.VectorDistance(t.Colfloat32,
EF.Functions.ToVector(t.Colint8, 3, "FLOAT32", "*"), "cosine")
               select t.Id);
foreach (var row in query4D)
{
    Console.WriteLine(row);
}

//LINQ - USING SPARSE VECTORS
//Below LINQ selects the ID from TABSPARSE table ordering the results based
on the vector_distance of COLINT8 column and "[3, [0, 1, 2], [3, 2, 1]]" sparse vector
data.
/* Generated query-
SELECT "t"."ID"
FROM "TABSPARSE" "t"
ORDER BY VECTOR_DISTANCE("t"."COLINT8",
TO_VECTOR('[3, [0, 1, 2], [3, 2, 1]]', 3, INT8, SPARSE), cosine)
*/
var query1S = (from t in ctx.Tabsparses
               orderby EF.Functions.VectorDistance(t.Colint8,
EF.Functions.ToVector("[3, [0, 1, 2], [3, 2, 1]]", 3, "INT8", "SPARSE"), "cosine")
               select t.Id);
foreach (var row in query1S)
{
    Console.WriteLine(row);
}

```

```

        //Below LINQ is similar to above LINQ but orders the results based on
        the vector_distance of COLFLOAT64 column and "[3, 2, 1]" vector data.
        /* Generated query-
        SELECT "t"."ID"
        FROM "TABSPARSE" "t"
        ORDER BY VECTOR_DISTANCE("t"."COLFLOAT64",
TO_VECTOR(:key1_1, 3, FLOAT64, SPARSE), cosine)
        */
        var query2S = (from t in ctx.Tabsparses
            orderby EF.Functions.VectorDistance(t.Colfloat64,
EF.Functions.ToVector(key1, 3, "FLOAT64", "SPARSE"), "cosine")
            select t.Id);
        foreach (var row in query2S)
        {
            Console.WriteLine(row);
        }

        //Similar LINQ as above but it uses an object of ODP.NET
        OracleVector Type instead of primitive type in ToVector EF function.
        /* Generated query-
        SELECT "t"."ID"
        FROM "TABSPARSE" "t"
        ORDER BY VECTOR_DISTANCE("t"."COLFLOAT64", TO_VECTOR(:keyV_1, 3, FLOAT64,
SPARSE), cosine)
        */
        var query3S = (from t in ctx.Tabsparses
            orderby EF.Functions.VectorDistance(t.Colfloat64,
EF.Functions.ToVector(keyV, 3, "FLOAT64", "SPARSE"), "cosine")
            select t.Id);
        foreach (var row in query3S)
        {
            Console.WriteLine(row);
        }

        //Below LINQ uses "*" as the storage format in which case the output
        vector format of TO_VECTOR is determined by the input vector argument format.
        /* Generated query-
        SELECT "t"."ID"
        FROM "TABSPARSE" "t"
        ORDER BY VECTOR_DISTANCE("t"."COLFLOAT32", TO_VECTOR("t"."COLINT8", 3,
FLOAT32, *), cosine)
        */
        var query4S = (from t in ctx.Tabsparses
            orderby EF.Functions.VectorDistance(t.Colfloat32,
EF.Functions.ToVector(t.Colint8, 3, "FLOAT32", "*"), "cosine")
            select t.Id);
        foreach (var row in query4S)
        {
            Console.WriteLine(row);
        }
    }
}

public partial class ModelContext : DbContext
{
    public ModelContext()
    {
    }

    public ModelContext(DbContextOptions<ModelContext> options)

```

```

        : base(options)
    {
    }

    public virtual DbSet<Tabdense> Tabdenses { get; set; }

    public virtual DbSet<Tabspare> Tabsparses { get; set; }

    //Enter your ODP.NET connection string.
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        => optionsBuilder.UseOracle("<CONNECTION STRING>");

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Tabdense>(entity =>
        {
            entity.HasKey(e => e.Id);

            entity.ToTable("TABDENSE");

            entity.Property(e => e.Id)
                .HasPrecision(10)
                .ValueGeneratedNever()
                .HasColumnName("ID");
            entity.Property(e => e.Colflex)
                .HasColumnType("VECTOR(*, *, DENSE)")
                .HasColumnName("COLFLEX");
            entity.Property(e => e.Colfloat32)
                .HasColumnType("VECTOR(3, FLOAT32, DENSE)")
                .HasColumnName("COLFLOAT32");
            entity.Property(e => e.Colfloat64)
                .HasColumnType("VECTOR(3, FLOAT64, DENSE)")
                .HasColumnName("COLFLOAT64");
            entity.Property(e => e.Colint8)
                .HasColumnType("VECTOR(3, INT8, DENSE)")
                .HasColumnName("COLINT8");
        });

        modelBuilder.Entity<Tabspare>(entity =>
        {
            entity.HasKey(e => e.Id);

            entity.ToTable("TABSPARSE");

            entity.Property(e => e.Id)
                .HasPrecision(10)
                .ValueGeneratedNever()
                .HasColumnName("ID");
            entity.Property(e => e.Colflex)
                .HasColumnType("VECTOR(*, *, SPARSE)")
                .HasColumnName("COLFLEX");
            entity.Property(e => e.Colfloat32)
                .HasColumnType("VECTOR(3, FLOAT32, SPARSE)")
                .HasColumnName("COLFLOAT32");
            entity.Property(e => e.Colfloat64)
                .HasColumnType("VECTOR(3, FLOAT64, SPARSE)")
                .HasColumnName("COLFLOAT64");
            entity.Property(e => e.Colint8)
                .HasColumnType("VECTOR(3, INT8, SPARSE)")
                .HasColumnName("COLINT8");
        });
    }

```

```
        OnModelCreatingPartial(modelBuilder);
    }

    partial void OnModelCreatingPartial(ModelBuilder modelBuilder);
}

public partial class Tabdense
{
    public int Id { get; set; }
    public OracleVector? Colint8 { get; set; }
    public OracleVector? Colfloat32 { get; set; }
    public OracleVector? Colfloat64 { get; set; }
    public OracleVector? Colflex { get; set; }
}

public partial class Tabsparse
{
    public int Id { get; set; }
    public OracleSparseVector? Colint8 { get; set; }
    public OracleSparseVector? Colfloat32 { get; set; }
    public OracleSparseVector? Colfloat64 { get; set; }
    public OracleSparseVector? Colflex { get; set; }
}
}
```

Spatial Data and NetTopologySuite

The Oracle database can store spatial data that represents physical locations and object shapes. ODP.NET can operate on Oracle spatial data natively.

EF Core maps to spatial data types with the NetTopologySuite (NTS) spatial library. Starting with Oracle EF Core 10, Oracle database SDO_GEOMETRY spatial data types can be accessed and mapped using the Oracle EF Core NTS library.

This library enables Oracle spatial columns to be mapped to .NET geometry types, such as Point and Polygon. LINQ statements for the Oracle database can include spatial logic, such as distance and intersection. EF Core apps can now use a standard way to query, insert, update, and delete Oracle database spatial data.

① See Also

- *Oracle AI Database Spatial Developer's Guide*
- [NetTopologySuite](#)
- [EF Core Spatial Data](#)

Setup

This Oracle spatial library is available in the Oracle.EntityFrameworkCore.NetTopologySuite package on NuGet Gallery. It is currently in preview. The NuGet package manager automatically retrieves other NTS packages needed to support Oracle EF Core spatial types.

To setup an app that uses Oracle spatial data in EF Core, follow these steps:

1. Add the Oracle.EntityFrameworkCore.NetTopologySuite package to your .NET project.

2. Use DbContext to enable the library. The extension method, `UseNetTopologySuite()`, configures the spatial provider settings, including the spatial tolerance level which controls the allowable geometric calculation error margin. For example:

```
services.AddDbContext<MyContext>(options =>
    options.UseOracle("<CONNECTION STRING>",
        o => o.UseNetTopologySuite(tolerance: 0.01));
```

3. In the entity model, define the spatial class. For example:

```
public class MySpatialEntity
{
    public int Id { get; set; }

    // NetTopologySuite.Geometries.Point
    public Point Location { get; set; }
}
```

4. Write a LINQ for the spatial data, such as

```
Point myPoint = new Point(10, 20);
var nearby = context.MySpatialEntities
    .Where(e => e.Location.IsWithinDistance(myPoint, 1000))
    .ToList();
```

Features

Oracle EF Core supports the following features:

- Automatically map Oracle `SDO_GEOMETRY` column data to NTS .NET data types, including
 - Point
 - LineString
 - Polygon
 - MultiPoint
 - MultiLineString
 - MultiPolygon
 - GeometryCollection
- Execute DML operations on NTS .NET types.
- Translate LINQ spatial operations to Oracle SQL spatial functions.
- EF Core migrations and database scaffolding with Oracle spatial columns
- Tolerance configuration in the `UseNetTopologySuite()` extension method.
 - If tolerance is not specified, then 0.005 is set as the default value.

Note

Oracle Spatial supports a superset of features not available or with limited features in NTS, such as curved geometry. EF Core spatial support is limited to NTS capabilities.

Oracle NetTopologySuite Members

Table 5-6 NetTopologySuite Methods and Oracle Spatial Functions Mapping

NetTopologySuite Methods	Oracle Spatial Functions
EnvelopeCombiner.CombineAsGeometry(geom[])	SDO_AGGR_MBR
Geometry.AsBinary()	SDO_UTIL.TO_WKBGEOMETRY
Geometry.AsText()	SDO_UTIL.TO_WKTGEOMETRY
Geometry.Buffer(distance)	SDO_GEOM.SDO_BUFFER
Geometry.Contains(geom)	SDO_CONTAINS
Geometry.ConvexHull()	SDO_GEOM.SDO_CONVEXHULL
Geometry.CoveredBy(geom)	SDO_RELATE
Geometry.Covers(geom)	SDO_RELATE
Geometry.Crosses(geom)	SDO_RELATE
Geometry.Disjoint(geom)	SDO_RELATE
Geometry.Difference(geom)	SDO_GEOM.DIFFERENCE
Geometry.Distance(geom)	SDO_GEOM.SDO_DISTANCE
Geometry.EqualsExact(geom)	SDO_RELATE
Geometry.EqualsTopologically(geom)	SDO_RELATE
Geometry.GetGeometryN(n)	SDO_UTIL.EXTRACT
Geometry.Intersection(geom)	SDO_GEOM.SDO_INTERSECTION
Geometry.Intersects(geom)	SDO_RELATE
Geometry.IsWithinDistance(geom, dist)	SDO_WITHIN_DISTANCE
Geometry.Overlaps(geom)	SDO_RELATE
Geometry.Relate(geom, pattern). Instead of the DE-9IM pattern, Oracle uses relevant mask words	SDO_RELATE
Geometry.Reverse() for LineString only and requires Oracle Database 23ai or higher.	SDO_UTIL.REVERSE_LINESTRING
Geometry.SymmetricDifference(geom)	SDO_GEOM.SDO_XOR
Geometry.ToBinary()	SDO_UTIL.TO_WKBGEOMETRY
Geometry.ToText()	SDO_UTIL.TO_WKTGEOMETRY
Geometry.Touches(geom)	SDO_RELATE
Geometry.Within(geom)	SDO_CONTAINS
Geometry.Union()	SDO_GEOM.SDO_SELF_UNION
Geometry.Union(geom)	SDO_GEOM.SDO_UNION
GeometryCombiner.Combine(geom[])	SDO_AGGR_UNION
ConvexHull.Create(geom[])	SDO_AGGR_CONVEXHULL
UnaryUnionOp.Union(geom[])	SDO_AGGR_UNION

Table 5-7 NetTopologySuite Properties and Oracle Spatial Functions Mapping

NetTopologySuite Properties	Oracle Spatial Functions
Geometry.Area	SDO_GEOM.SDO_AREA
Geometry.Centroid	SDO_GEOM.SDO_CENTROID
Geometry.Envelope	SDO_GEOM.SDO_MBR
Geometry.GeometryType	CASE on SDO_GTYPE
Geometry.InteriorPoint	SDO_GEOM.SDO_POINTONSURFACE
Geometry.IsEmpty	Composite null checks on SDO_GEOMETRY fields
Geometry.IsSimple	SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT
Geometry.IsValid	SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT
Geometry.Length	SDO_GEOM.SDO_LENGTH
Geometry.NumGeometries	SDO_UTIL.GETNUMELEM
Geometry.NumPoints	SDO_UTIL.GETNUMVERTICES
Geometry.OgcGeometryType	CASE on SDO_GTYPE
Geometry.PointOnSurface	SDO_GEOM.SDO_POINTONSURFACE
Geometry.SRID	SDO_SRID(field)
GeometryCollection.Count	SDO_UTIL.GETNUMELEM
LineString.Count	SDO_UTIL.GETNUMVERTICES
LineString.EndPoint	SDO_LRS.GEOM_SEGMENT_END_PT
LineString.IsClosed	SDO_EQUAL(SDO_LRS.GEOM_SEGMENT_START_PT and SDO_LRS.GEOM_SEGMENT_END_PT)
LineString.IsRing	SDO_EQUAL(SDO_LRS.GEOM_SEGMENT_START_PT, SDO_LRS.GEOM_SEGMENT_END_PT), and SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT
LineString.StartPoint	SDO_LRS.GEOM_SEGMENT_START_PT
Point.X	SDO_POINT.X
Point.Y	SDO_POINT.Y
Point.Z	SDO_POINT.Z
Polygon.ExteriorRing	SDO_UTIL.POLYGONTOLINE and SDO_UTIL.EXTRACT(geom, 1, 0)
Polygon.NumInteriorRings	SDO_UTIL.GETNUMRINGS - 1

Limitations and Workarounds

- M-values (Measure Coordinates): Oracle EF Core with NTS does not support M-value (measure) coordinates. Geometries with M or ZM ordinates have the M-value dropped during serialization. It is not included in the generated SDO_GEOMETRY. When reading geometries from Oracle Spatial, M values are not materialized into the corresponding NTS geometry objects.

Workaround:

For M-aware or LRS (Linear Referencing System) operations, use direct Oracle SQL or stored procedures:

Example 1: Reading M-value geometries

SDO_GEOMETRY object in the Oracle Database containing geometries with the following M values:

```
SDO_GEOMETRY(
    3303,          -- LineString with Z and M
    NULL,
    NULL,
    SDO_ELEM_INFO_ARRAY(1,2,1),
    SDO_ORDINATE_ARRAY(10,20,5,100, 30,40,10,200)
)
```

Oracle EF Core code to read the SDO_GEOMETRY object in the Oracle Database and its output:

```
// C#
var segment = context.RoadSegments.First();
var coords = segment.Geometry.Coordinates;

// output:
Coordinate[0] = (10, 20, 5)
Coordinate[1] = (30, 40, 10)
```

The M-values (100, 200) are dropped. They do not appear in the resulting objects.

Example 2: Writing M-value geometries

Oracle EF Core code to update geometries with M values:

```
// C#
// EF Model
public class RoadSegment
{
    public int Id { get; set; }
    public LineString Geometry { get; set; }
}

// SaveChanges Code:
var segment = new RoadSegment
{
    Geometry = new LineString(new[]
    {
        new CoordinateZM(10, 20, 5, 100),
        new CoordinateZM(30, 40, 10, 200)
    })
};

context.RoadSegments.Add(segment);
context.SaveChanges();
```

Executing Oracle EF Core code to update geometries with M values results in the following SDO_GEOMETRY object in the Oracle Database:

```
SDO_GEOMETRY(
  2003,          -- 2D LineString type (M is ignored)
  NULL,
  NULL,
  SDO_ELEM_INFO_ARRAY(1,2,1),
  SDO_ORDINATE_ARRAY(10,20,5, 30,40,10)
)
```

- **Relate() method:** The `Relate()` method in NTS checks spatial relationships between geometries using the Dimensionally Extended Nine-Intersection Model (DE-9IM) matrix. Oracle Spatial uses a different approach known as topological relationship masks, rather than the DE-9IM patterns used by other spatial libraries. When you use the `Relate()` method in Entity Framework Core LINQ queries with a DE-9IM pattern (the standard NTS usage), these patterns will not translate correctly to Oracle SQL. Queries using the DE-9IM pattern may fail or return incorrect results.

Workaround

Instead of the DE-9IM matrix, provide an Oracle Spatial relationship mask string as the second parameter to the `Relate()` method. The Oracle EF Core NTS library inserts this string as the spatial relationship mask in the `SDO_RELATE` mask parameter during SQL translation.

Example

LINQ query with `.Relate()` and Oracle mask.

The following query uses the `Relate()` method to check the spatial relationship between each row's geometry and the target geometry, specifying the Oracle spatial relationship mask `TOUCH` instead of a DE-9IM pattern.

```
// C#
var targetGeometry = factory.CreatePolygon(new[]
{
    new Coordinate(0, 0),
    new Coordinate(10, 0),
    new Coordinate(10, 10),
    new Coordinate(0, 10),
    new Coordinate(0, 0)
});

var result = context.GeometryEntities
    .Where(e => e.Geometry.Relate(targetGeometry, "TOUCH"))
    .Select(e => new { e.Id, e.Geometry })
    .ToList();
```

Oracle EF Core translates this LINQ expression (LINQ query with `.Relate()` and Oracle mask) into the following Oracle SQL:

```
SELECT e."Id", e."Geometry"
FROM "GeometryEntities" e
WHERE SDO_RELATE(e."Geometry", :targetGeometry, 'mask=TOUCH') = 'TRUE'
```

Oracle supports combining multiple relationship masks using the logical OR operator (+).

Breaking Changes

Application Programming Interface Changes

- Oracle EF Core UseOracleSQLCompatibility extension method

In EF Core 6 and higher versions with Oracle EF Core version 23, UseOracleSQLCompatibility extension method now takes an enumeration type as an argument rather than a string value. This extension method's default value will be the same as the Oracle EF Core version. For example, if you are using Oracle EF Core 23, then its value will be `OracleSQLCompatibility.DatabaseVersion23`.

This default assumes the app is using the same database version as the Oracle EF Core version, that is, Oracle AI Database 26ai with Oracle 23ai client. If they are not the same version, then the developer should modify the enumeration value to match the database version they are connecting to. This ensures fewer compatibility errors and allows Oracle EF Core to use the latest Oracle AI Database features.

See Also

[UseOracleSQLCompatibility\(enum version\)](#)

Desupported APIs

The desupported APIs are as follows:

- `ModelBuilder.UseOracleIdentityColumn` Extension Method
Starting with EF Core 5, ODP.NET EF Core desupports `UseOracleIdentityColumn`. Developers should call the `ModelBuilder.UseIdentityColumn` extension method instead, which provides identical functionality. `UseOracleIdentityColumn` remains supported for EF Core 3.1 and earlier releases.
- `ModelBuilder` and `PropertyBuilder` `ForOracleUseSequenceHiLo` Extension Method
Starting with EF Core 5, ODP.NET EF Core desupports `ForOracleUseSequenceHiLo`. Developers should call the `ModelBuilder.UseHiLo` extension method instead, which provides identical functionality. `ForOracleUseSequenceHiLo` remains supported for EF Core 3.1 and earlier releases.
- `Oracle.EntityFrameworkCore.Migrations.Operations.OracleCreateUserOperation` Class
Starting with Oracle EF Core 21c, the `OracleCreateUserOperation` class is desupported. Developers should call the `MigrationBuilder.Sql` method instead with a `CREATE USER` SQL statement to create a user if they are using the desupported API directly.
- `Oracle.EntityFrameworkCore.Migrations.Operations.OracleDropUserOperation` Class
Starting with Oracle EF Core 21c, the `OracleDropUserOperation` class is desupported. Developers should call the `MigrationBuilder.Sql` method instead with a `DROP USER` SQL statement to drop a user if they are using the desupported API directly.

6

Oracle Data Provider for .NET Classes

This chapter describes the following Oracle Data Provider for .NET classes.

- [OracleAccessToken Class](#)
- [OracleClientFactory Class](#)
- [OracleCommand Class](#)
- [OracleCommandBuilder Class](#)
- [OracleConnection Class](#)
- [OracleConfiguration Class](#)
- [OracleConfigurationProvider Class](#)
- [OracleConnectionOpenEventArgs Class](#)
- [OracleConnectionOpenEventHandler Delegate](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleCredential Class](#)
- [OracleDataAdapter Class](#)
- [OracleDatabase Class](#)
- [OracleDataReader Class](#)
- [OracleDataSource Class](#)
- [OracleDataSourceBuilder Class](#)
- [OracleDataSourceCollection Class](#)
- [OracleDataSourceCommand Class](#)
- [OracleDataSourceEnumerator Class](#)
- [OracleDBMSAssert Class](#)
- [OracleError Class](#)
- [OracleErrorCollection Class](#)
- [OracleException Class](#)
- [OracleInfoMessageEventArgs Class](#)
- [OracleInfoMessageEventHandler Delegate](#)
- [OracleLogicalTransaction Class](#)
- [OracleOnsServerCollection Class](#)
- [OracleOpaqueString Class](#)
- [OracleParameter Class](#)
- [OracleParameterCollection Class](#)
- [OraclePermission Class](#)

- [OracleRefreshAccessTokenEventArgs Class](#)
- [OraclePermissionAttribute Class](#)
- [OracleRefreshAccessTokenEventArgs Class](#)
- [OracleRoleCollection Class](#)
- [OracleRowUpdatedEventArgs Class](#)
- [OracleRowUpdatedEventHandler Delegate](#)
- [OracleRowUpdatingEventArgs Class](#)
- [OracleRowUpdatingEventHandler Delegate](#)
- [OracleSessionlessTransactionOptions Class](#)
- [OracleShardingKey Class](#)
- [OracleSparseVector Class](#)
- [OracleTransaction Class](#)
- [OracleVector Class](#)
- [OracleAllowedLogonVersionClient Enumeration](#)
- [OracleCollectionType Enumeration](#)
- [OracleConnectionOpenReason Enumeration](#)
- [OracleConnectionType Enumeration](#)
- [OracleDBAPrivilege Enumeration](#)
- [OracleDBShutdownMode Enumeration](#)
- [OracleDBStartupMode Enumeration](#)
- [OracleDbType Enumeration](#)
- [OracleDRCPPurity Enumeration](#)
- [OracleIdentityType Enumeration](#)
- [OracleParameterStatus Enumeration](#)
- [OraclePasswordAuth Enumeration](#)
- [OracleSessionlessTransactionStartBehavior Enumeration](#)
- [OracleTokenAuth Enumeration](#)
- [OracleVectorType Enumeration](#)

OracleAccessToken Class

An `OracleAccessToken` object stores Oracle Identity and Access Management(IAM) and Microsoft Entra ID OAuth information required for token authentication.

Class Inheritance

`System.Object`

`Oracle.DataAccess.OracleAccessToken`

Declaration

```
// C#
public sealed class OracleAccessToken
```

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

Example 6-1 Using OracleAccessToken Constructor and Event

```
// C#
public class OracleAccessToken
{
    //Constructor for OAuth
    public OracleAccessToken(char[] token);

    // Constructor
    public OracleAccessToken(char[] dbToken, char[] privateKey);

    // Event to get refreshed token before expiry
    public event OracleRefreshAccessTokenEventHandler RefreshAccessToken;
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAccessToken Members](#)
- [OracleAccessToken Constructors](#)
- [OracleAccessToken Event](#)

OracleAccessToken Members

OracleAccessToken members are listed in the following tables.

OracleAccessToken Constructor

OracleAccessToken constructors are listed in [OracleAccessToken Constructors](#).

Table 6-1 OracleAccessToken Constructor

Constructor	Description
OracleAccessToken Constructors	OracleAccessToken constructors instantiate new instances of OracleAccessToken class.

OracleAccessToken Event

OracleAccessToken event is listed in [OracleAccessToken Event](#).

Table 6-2 OracleAccessToken Event

Event	Description
RefreshAccessToken Event	Triggered 60 seconds before the signature token expiration time so that application can provide a refreshed token

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAccessToken Class](#)

OracleAccessToken Constructors

OracleAccessToken constructors instantiate new instances of OracleAccessToken class.

Overload List:

- [OracleAccessToken\(char\)](#)
This constructor creates an OracleAccessToken object with the access token provided by the application.
- [OracleAccessToken\(char, char\)](#)
This constructor creates an OracleAccessToken object with the signature token and private key provided by the application. Only Signature token type will be supported through this class.

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAccessToken Class](#)
- [OracleAccessToken Members](#)

OracleAccessToken(char)

This constructor creates an OracleAccessToken object with the access token provided by the application.

Declaration

```
// C#
public OracleAccessToken(char[] token);
```


Exceptions

`ArgumentNullException` is raised if token is passed in as null.

Description

The `char[]` passed by the application for the access token will be cleared by ODP.NET. Applications should not depend on this `char[]` once this constructor is called.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAccessToken Class](#)
- [OracleAccessToken Members](#)

OracleAccessToken(char, char)

This constructor creates an `OracleAccessToken` object with the signature token and private key provided by the application. Only Signature token type will be supported through this class.

Declaration

```
// C#  
public OracleAccessToken(char[] dbToken, char[] privateKey);
```

Exceptions

- `ArgumentNullException` is raised if any parameters are passed in as null.
- Argument exception is thrown if the provided token is invalid.

Description

The `char[]` passed by the application for the signature token and private key will be cleared by ODP.NET. Applications should not depend on these `char[]` once this constructor is called.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAccessToken Class](#)
- [OracleAccessToken Members](#)

RefreshAccessToken Event

This event is triggered 60 seconds before the database token expiration time so that application can provide a refreshed token.

Declaration

```
// C#  
public event OracleRefreshAccessTokenEventHandler RefreshAccessToken;
```

Description

The event handler receives an `OracleRefreshAccessTokenEventArgs` object in which the application should set the refreshed database token and private key, or access token. The event handler is not called unless a connection has been made with the token, and the token has been found to be valid.

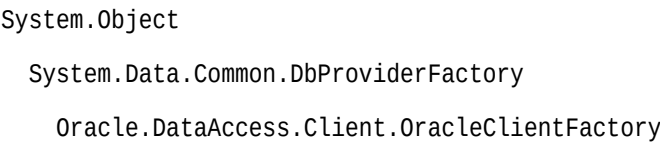
See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAccessToken Class](#)
- [OracleAccessToken Members](#)

OracleClientFactory Class

An `OracleClientFactory` object allows applications to instantiate ODP.NET classes in a generic way.

Class Inheritance



Declaration

```
// C#
public sealed class OracleClientFactory : DbProviderFactory
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET Core	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#

using System;
using System.Data;
using System.Data.Common;
using Oracle.DataAccess.Client;

class FactorySample
{
    static void Main()
    {
        string constr = "user id=scott;password=tiger;data source=oracle";

        DbProviderFactory factory =
            DbProviderFactories.GetFactory("Oracle.DataAccess.Client");

        DbConnection conn = factory.CreateConnection();

        try
        {
            conn.ConnectionString = constr;
            conn.Open();

            DbCommand cmd = factory.CreateCommand();
            cmd.Connection = conn;
            cmd.CommandText = "select * from emp";

            DbDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
                Console.WriteLine(reader["EMPNO"] + " : " + reader["ENAME"]);
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
            Console.WriteLine(ex.StackTrace);
        }
    }
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleClientFactory Members](#)
- [OracleClientFactory Field](#)
- [OracleClientFactory Constructor](#)
- [OracleClientFactory Public Properties](#)
- [OracleClientFactory Public Methods](#)

OracleClientFactory Members

OracleClientFactory members are listed in the following tables.

OracleClientFactory Field

The OracleClientFactory field is listed in [Table 6-3](#)

Table 6-3 OracleClientFactory Field

Property	Description
Instance	Gets an instance of the OracleClientFactory class

OracleClientFactory Constructor

The OracleClientFactory constructor is listed in [Table 6-4](#)

Table 6-4 OracleClientFactory Constructor

Property	Description
OracleClientFactory Constructor	Instantiates a new instance of OracleClientFactory class

OracleClientFactory Public Properties

The OracleClientFactory public properties are listed in [Table 6-5](#).

Table 6-5 OracleClientFactory Public Properties

Property	Description
CanCreateDataSourceEnumerator	Indicates whether or not the CreateDataSourceEnumerator method is supported

OracleClientFactory Public Methods

OracleClientFactory Public Methods are listed in [Table 6-6](#).

Table 6-6 OracleClientFactory Public Method

Method	Description
CreateCommand	Returns a DbCommand object that represents an OracleCommand object
CreateCommandBuilder	Returns a DbCommandBuilder object that represents an OracleCommandBuilder object
CreateConnection	Returns a DbConnection object that represents an OracleConnection object
CreateConnectionStringBuilder	Returns a DbConnectionStringBuilder object that represents an OracleConnectionStringBuilder object
CreateDataAdapter	Returns a DbDataAdapter object that represents an OracleDataAdapter object

Table 6-6 (Cont.) OracleClientFactory Public Method

Method	Description
CreateDataSource	Creates a new OracleDataSource object using the provided connection string <i>Not available in ODP.NET, Unmanaged Driver and ODP.NET, Managed Driver</i>
CreateDataSourceEnumerator	Returns a DbDataSourceEnumerator object that represents an OracleDataSourceEnumerator object
CreateParameter	Returns a DbParameter object that represents an OracleParameter object
CreatePermission	Returns a CodeAccessPermission object that represents an OraclePermission object

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleClientFactory Class](#)

OracleClientFactory Field

The OracleClientFactory field is listed in [Table 6-7](#)

Table 6-7 OracleClientFactory Field

Property	Description
Instance	Gets an instance of the OracleClientFactory class

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleClientFactory Class](#)
- [OracleClientFactory Members](#)

Instance

The Instance field gets an instance of the OracleClientFactory class. This can be used to retrieve strongly typed data objects.

Declaration

```
// C#  
public static readonly OracleClientFactory Instance
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleClientFactory Class](#)
- [OracleClientFactory Members](#)

OracleClientFactory Constructor

The OracleClientFactory constructor creates a new instances of the OracleClientFactory class.

Declaration

```
// C#
public OracleClientFactory();
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleClientFactory Class](#)
- [OracleClientFactory Members](#)

OracleClientFactory Public Properties

The OracleClientFactory public properties are listed in [Table 6-8](#).

Table 6-8 OracleClientFactory Public Properties

Property	Description
CanCreateDataSourceEnumerator	Indicates whether or not the CreateDataSourceEnumerator method is supported

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleClientFactory Class](#)
- [OracleClientFactory Members](#)

CanCreateDataSourceEnumerator

This property indicates whether or not the CreateDataSourceEnumerator method is supported.

Declaration

```
// C#
public override bool CanCreateDataSourceEnumerator { get; }
```

Property Value

Returns true.

Remarks

ODP.NET supports the `OracleDataSourceEnumerator` object.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleClientFactory Class](#)
- [OracleClientFactory Members](#)

OracleClientFactory Public Methods

The `OracleClientFactory` public method is listed in [Table 6-9](#).

Table 6-9 OracleClientFactory Public Method

Method	Description
CreateCommand	Returns a <code>DbCommand</code> object that represents an <code>OracleCommand</code> object
CreateCommandBuilder	Returns a <code>DbCommandBuilder</code> object that represents an <code>OracleCommandBuilder</code> object
CreateConnection	Returns a <code>DbConnection</code> object that represents an <code>OracleConnection</code> object
CreateConnectionStringBuilder	Returns a <code>DbConnectionStringBuilder</code> object that represents an <code>OracleConnectionStringBuilder</code> object
CreateDataAdapter	Returns a <code>DbDataAdapter</code> object that represents an <code>OracleDataAdapter</code> object
CreateDataSource	Creates a new <code>OracleDataSource</code> object using the provided connection string <i>Not available in ODP.NET, Unmanaged Driver and ODP.NET, Managed Driver</i>
CreateDataSourceEnumerator	Returns a <code>DbDataSourceEnumerator</code> object that represents an <code>OracleDataSourceEnumerator</code> object
CreateParameter	Returns a <code>DbParameter</code> object that represents an <code>OracleParameter</code> object
CreatePermission	Returns a <code>CodeAccessPermission</code> object that represents an <code>OraclePermission</code> object

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleClientFactory Class](#)
- [OracleClientFactory Members](#)

CreateCommand

This method returns a DbCommand object that represents an OracleCommand object.

Declaration

```
// C#  
public override DbCommand CreateCommand();
```

Return Value

A DbCommand object that represents an OracleCommand object.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleClientFactory Class](#)
- [OracleClientFactory Members](#)

CreateCommandBuilder

This method returns a DbCommandBuilder object that represents an OracleCommandBuilder object.

Declaration

```
// C#  
public override DbCommandBuilder CreateCommandBuilder();
```

Return Value

A DbCommandBuilder object that represents an OracleCommandBuilder object.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleClientFactory Class](#)
- [OracleClientFactory Members](#)

CreateConnection

This method returns a DbConnection object that represents an OracleConnection object.

Declaration

```
// C#  
public override DbConnection CreateConnection();
```

Return Value

A DbConnection object that represents an OracleConnection object.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleClientFactory Class](#)
- [OracleClientFactory Members](#)

CreateConnectionStringBuilder

This method returns a DbConnectionStringBuilder object that represents an OracleConnectionStringBuilder object.

Declaration

```
// C#  
public override DbConnectionStringBuilder CreateConnectionStringBuilder();
```

Return Value

A DbConnectionStringBuilder object that represents an OracleConnectionStringBuilder object.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleClientFactory Class](#)
- [OracleClientFactory Members](#)

CreateDataAdapter

This method returns a DbDataAdapter object that represents an OracleDataAdapter object.

Declaration

```
// C#  
public override DbDataAdapter CreateDataAdapter();
```

Return Value

A DbDataAdapter object that represents an OracleDataAdapter object.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleClientFactory Class](#)
- [OracleClientFactory Members](#)

CreateDataSource

This method creates a new OracleDataSource object using the provided connection string.

Declaration

```
// C#  
OracleDataSource CreateDataSource(string connectionString)
```

Parameters

- *connectionString*
The connection string to connect to the database

Return Value

An OracleDataSource object ready to create database connections and commands.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleClientFactory Class](#)
- [OracleClientFactory Members](#)
- ["OracleDataSourceEnumerator Class"](#)

CreateDataSourceEnumerator

This method returns a DbDataSourceEnumerator object that represents an OracleDataSourceEnumerator object.

Declaration

```
// C#  
public override DbDataSourceEnumerator CreateDataSourceEnumerator();
```

Return Value

A DbDataSourceEnumerator object that represents an OracleDataSourceEnumerator object.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleClientFactory Class](#)
- [OracleClientFactory Members](#)
- ["OracleDataSourceEnumerator Class"](#)

CreateParameter

This method returns a DbParameter object that represents an OracleParameter object.

Declaration

```
// C#  
public override DbParameter CreateParameter();
```

Return Value

A DbParameter object that represents an OracleParameter object.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleClientFactory Class](#)
- [OracleClientFactory Members](#)

CreatePermission

This method returns a CodeAccessPermission object that represents an OraclePermission object.

Declaration

```
// C#  
public override System.Security.CodeAccessPermission CreatePermission(  
    System.Security.Permissions.PermissionState state);
```

Parameter

- `state`
A `PermissionState` object.

Return Value

A `CodeAccessPermission` object that represents an `OraclePermission` object.

Remarks

This method enables users, writing provider-independent code, to get a `CodeAccessPermission` instance that represents an `OraclePermission` object.

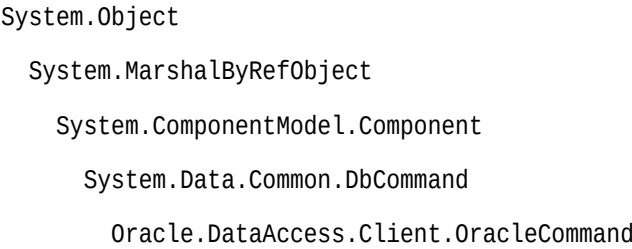
 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleClientFactory Class](#)
- [OracleClientFactory Members](#)

OracleCommand Class

An `OracleCommand` object represents a SQL command, a stored procedure, or a table name. The `OracleCommand` object is responsible for formulating the request and passing it to the database. If results are returned, `OracleCommand` is responsible for returning results as an `OracleDataReader`, a .NET `XmlReader`, a .NET `Stream`, a scalar value, or as output parameters.

Class Inheritance



Declaration

```
// C#
public sealed class OracleCommand : DbCommand, ICloneable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

The execution of any transaction-related statements from an `OracleCommand` is not recommended because it is not reflected in the state of the `OracleTransaction` object represents the current local transaction, if one exists.

`ExecuteXmlReader`, `ExecuteStream`, and `ExecuteToStream` methods are only supported for XML operations.

`ExecuteReader` and `ExecuteScalar` methods are not supported for XML operations.

To minimize the number of open server cursors, `OracleCommand` objects should be explicitly disposed.

Example

```
// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;

class OracleCommandSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        string cmdQuery = "select ename, empno from emp";

        // Create the OracleCommand
        OracleCommand cmd = new OracleCommand(cmdQuery);

        cmd.Connection = con;
        cmd.CommandType = CommandType.Text;

        // Execute command, create OracleDataReader object
        OracleDataReader reader = cmd.ExecuteReader();

        while (reader.Read())
        {
            // output Employee Name and Number
        }
    }
}
```

```
        Console.WriteLine("Employee Name : " + reader.GetString(0) + " , " +  
            "Employee Number : " + reader.GetDecimal(1));  
    }  
  
    // Clean up  
    reader.Dispose();  
    cmd.Dispose();  
    con.Dispose();  
}  
}
```

📘 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Members](#)
- [OracleCommand Constructors](#)
- [OracleCommand Static Methods](#)
- [OracleCommand Properties](#)
- [OracleCommand Public Methods](#)

OracleCommand Members

OracleCommand members are listed in the following tables.

OracleCommand Constructors

OracleCommand constructors are listed in [Table 6-10](#).

Table 6-10 OracleCommand Constructors

Constructor	Description
OracleCommand Constructors	Instantiates a new instance of OracleCommand class (Overloaded)

OracleCommand Static Methods

The OracleCommand static method is listed in [Table 6-11](#).

Table 6-11 OracleCommand Static Method

Method	Description
<code>Equals</code>	Inherited from <code>System.Object</code> (Overloaded)

OracleCommand Properties

OracleCommand properties are listed in [Table 6-12](#).

Table 6-12 OracleCommand Properties

Property	Description
AddRowid	Adds the ROWID as part of the select list
AddToStatementCache	Causes executed statements to be cached, when the property is set to <code>true</code> and statement caching is enabled
ArrayBindCount	Specifies if the array binding feature is to be used and also specifies the maximum number of array elements to be bound in the <code>Value</code> property
ArrayBindRowsAffected	Returns the number of affected rows for each iteration while executing a DML using array binding
BindByName	Specifies the binding method in the collection
CommandText	Specifies the SQL statement or stored procedure to run against the Oracle database or the XML data used to store changes to the Oracle database
CommandTimeout	Specifies the number of seconds the command is allowed to execute before terminating the execution with an exception
CommandType	Specifies the command type that indicates how the <code>CommandText</code> property is to be interpreted
Connection	Specifies the <code>OracleConnection</code> object that is used to identify the connection to execute a command
Container	Inherited from <code>System.ComponentModel.Component</code>
DesignTimeVisible	Specifies whether or not the <code>OracleCommand</code> object is visible on designer controls.
FetchSize	Specifies the size of <code>OracleDataReader</code> 's internal cache to store result set data
FetchSqlId	Indicates whether or not to fetch the <code>SQL_ID</code> of the executed statements <i>Not Available in ODP.NET, Unmanaged Driver</i>
ImplicitRefCursors	Specifies an array of <code>OracleRefCursors</code> mapped to an implicit resultset returned by the stored procedure.
InitialLOBFetchSize	Specifies the amount of data that the <code>OracleDataReader</code> initially fetches for LOB columns
InitialLONGFetchSize	Specifies the amount of data that the <code>OracleDataReader</code> initially fetches for LONG and LONG RAW columns
Notification	Indicates that there is a notification request for the command
NotificationAutoEnlist	Indicates whether or not to register for a continuous query notification with the database automatically when the command is executed
Parameters	Specifies the parameters for the SQL statement or stored procedure
RowSize	Specifies the amount of memory needed by the <code>OracleDataReader</code> internal cache to store one row of data
RowsToFetchPerRoundTrip	Specifies the total number of rows to retrieve per database round trip <i>Not Available in ODP.NET, Unmanaged Driver</i>
Site	Inherited from <code>System.ComponentModel.Component</code>

Table 6-12 (Cont.) OracleCommand Properties

Property	Description
SqlId	Provides the SQL_ID corresponding to the last executed statement on the OracleCommand <i>Not Available in ODP.NET, Unmanaged Driver</i>
Transaction	Specifies the OracleTransaction object in which the OracleCommand executes
UpdatedRowSource	Specifies how query command results are applied to the row being updated
UseEdmMapping	Indicates whether or not the command object utilizes the Entity Data Model mapping configuration values
XmlCommandType	Specifies the type of XML operation on the OracleCommand
XmlQueryProperties	Specifies the properties that are used when an XML document is created from the result set of a SQL query statement
XmlSaveProperties	Specifies the properties that are used when an XML document is used to save changes to the database

OracleCommand Public Methods

OracleCommand public methods are listed in [Table 6-13](#).

Table 6-13 OracleCommand Public Methods

Public Method	Description
Cancel	Attempts to cancel a command that is currently executing on a particular connection
Clone	Creates a copy of OracleCommand object
CreateObjRef	Inherited from System.MarshalByRefObject
CreateParameter	Creates a new instance of OracleParameter class
Dispose	Releases any resources or memory allocated by the object
Equals	Inherited from System.Object (Overloaded)
ExecuteNonQuery	Executes a SQL statement or a command using the XmlCommandType and CommandText properties and returns the number of rows affected
ExecuteNonQueryAsync	Returns a Task-based asynchronous version of ExecuteNonQuery() (Overloaded).
ExecuteReader	Executes a command (Overloaded)
ExecuteReaderAsync	Returns a Task-based asynchronous version of OracleCommand.ExecuteReader(), which fetches the result set as an OracleDataReader object. (Overloaded)
ExecuteScalar	Returns the first column of the first row in the result set returned by the query
ExecuteScalarAsync	Returns a Task-based asynchronous version of OracleCommand.ExecuteScalar(), which returns the first column of the first row in the result set returned by the query. (Overloaded)

Table 6-13 (Cont.) OracleCommand Public Methods

Public Method	Description
ExecuteStream	Executes a command using the <code>XmlCommandType</code> and <code>CommandText</code> properties and returns the results in a new <code>Stream</code> object
ExecuteToStream	Executes a command using the <code>XmlCommandType</code> and <code>CommandText</code> properties and appends the results as an XML document to the existing <code>Stream</code>
ExecuteXmlReader	Executes a command using the <code>XmlCommandType</code> and <code>CommandText</code> properties and returns the result as an XML document in a .NET <code>XmlTextReader</code> object
ExecuteXmlReaderAsync	Returns a Task-based asynchronous version of <code>OracleCommand.ExecuteXmlReader()</code> , which fetches the result set as an <code>XmlReader</code> object (Overloaded)
<code>GetHashCode</code>	Inherited from <code>System.Object</code>
<code>GetLifetimeService</code>	Inherited from <code>System.MarshalByRefObject</code>
<code>GetType</code>	Inherited from <code>System.Object</code>
<code>InitializeLifetimeService</code>	Inherited from <code>System.MarshalByRefObject</code>
Prepare	<i>This method is a no-op</i>
<code>ToString</code>	Inherited from <code>System.Object</code>

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)

OracleCommand Constructors

`OracleCommand` constructors instantiate new instances of `OracleCommand` class.

Overload List:

- [OracleCommand\(\)](#)
This constructor instantiates a new instance of `OracleCommand` class.
- [OracleCommand\(string\)](#)
This constructor instantiates a new instance of `OracleCommand` class using the supplied SQL command or stored procedure, and connection to the Oracle database.
- [OracleCommand\(string, OracleConnection\)](#)
This constructor instantiates a new instance of `OracleCommand` class using the supplied SQL command or stored procedure, and connection to the Oracle database.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

OracleCommand()

This constructor instantiates a new instance of `OracleCommand` class.

Declaration

```
// C#  
public OracleCommand();
```

Remarks

Default constructor.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

OracleCommand(string)

This constructor instantiates a new instance of `OracleCommand` class using the supplied SQL command or stored procedure, and connection to the Oracle database.

Declaration

```
// C#  
public OracleCommand(string cmdText);
```

Parameters

- *cmdText*
The SQL command or stored procedure to be executed.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

OracleCommand(string, OracleConnection)

This constructor instantiates a new instance of `OracleCommand` class using the supplied SQL command or stored procedure, and connection to the Oracle database.

Declaration

```
// C#
public OracleCommand(string cmdText, OracleConnection OracleConnection);
```

Parameters

- `cmdText`
The SQL command or stored procedure to be executed.
- `OracleConnection`
The connection to the Oracle database.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

OracleCommand Static Methods

The `OracleCommand` static method is listed in [Table 6-14](#).

Table 6-14 OracleCommand Static Method

Method	Description
<code>Equals</code>	Inherited from <code>System.Object</code> (Overloaded)

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

OracleCommand Properties

`OracleCommand` properties are listed in [Table 6-15](#).

Table 6-15 OracleCommand Properties

Property	Description
AddRowid	Adds the ROWID as part of the select list
AddToStatementCache	Causes executed statements to be cached, when the property is set to true and statement caching is enabled
ArrayBindCount	Specifies if the array binding feature is to be used and also specifies the maximum number of array elements to be bound in the Value property
ArrayBindRowsAffected	Returns the number of affected rows for each iteration while executing a DML using array binding
BindByName	Specifies the binding method in the collection
CommandText	Specifies the SQL statement or stored procedure to run against the Oracle database or the XML data used to store changes to the Oracle database
CommandTimeout	Specifies the number of seconds the command is allowed to execute before terminating the execution with an exception
CommandType	Specifies the command type that indicates how the CommandText property is to be interpreted
Connection	Specifies the OracleConnection object that is used to identify the connection to execute a command
Container	Inherited from System.ComponentModel.Component
DesignTimeVisible	Specifies whether or not the OracleCommand object is visible on designer controls.
FetchSize	Specifies the size of OracleDataReader's internal cache to store result set data
FetchSqlId	Indicates whether or not to fetch the SQL_ID of the executed statements <i>Not Available in ODP.NET, Unmanaged Driver</i>
ImplicitRefCursors	Specifies an array of OracleRefCursors mapped to an implicit resultset returned by the stored procedure.
InitialLOBFetchSize	Specifies the amount of data that the OracleDataReader initially fetches for LOB columns
InitialLONGFetchSize	Specifies the amount that of data the OracleDataReader initially fetches for LONG and LONG RAW columns
Notification	Indicates that there is a notification request for the command
NotificationAutoEnlist	Indicates whether or not to register for a continuous query notification with the database automatically when the command is executed
Parameters	Specifies the parameters for the SQL statement or stored procedure
RowSize	Specifies the amount of memory needed by the OracleDataReader internal cache to store one row of data
RowsToFetchPerRoundTrip	Specifies the total number of rows to retrieve per database round trip <i>Not Available in ODP.NET, Unmanaged Driver</i>
Site	Inherited from System.ComponentModel.Component

Table 6-15 (Cont.) OracleCommand Properties

Property	Description
SqlId	Provides the SQL_ID corresponding to the last executed statement on the OracleCommand
Transaction	Specifies the OracleTransaction object in which the OracleCommand executes <i>Not Available in ODP.NET, Unmanaged Driver</i>
UpdatedRowSource	Specifies how query command results are applied to the row being updated
UseEdmMapping	Indicates whether or not the command object utilizes the Entity Data Model mapping configuration values
XmlCommandType	Specifies the type of XML operation on the OracleCommand
XmlQueryProperties	Specifies the properties that are used when an XML document is created from the result set of a SQL query statement
XmlSaveProperties	Specifies the properties that are used when an XML document is used to save changes to the database

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

AddRowid

This property adds the ROWID as part of the select list.

Declaration

```
// C#  
public bool AddRowid {get; set;}
```

Property Value

bool

Remarks

Default is false.

This ROWID column is hidden and is not accessible by the application. To gain access to the ROWIDs of a table, the ROWID must explicitly be added to the select list without the use of this property.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)
- ["LOB Support"](#) for further information on how this property used with LOBs

AddToStatementCache

This property causes executed statements to be cached when the property is set to true and statement caching is enabled. If statement caching is disabled or if this property is set to false, the executed statement is not cached.

Declaration

```
// C#  
public bool AddToStatementCache{get; set;}
```

Return Value

Returns bool value. A value of true indicates that statements are being added to the cache, false indicates otherwise.

Property Value

A bool value that indicates that the statements will be cached when they are executed, if statement caching is enabled.

Remarks

Default is true.

AddToStatementCache is ignored if statement caching is disabled. Statement caching is enabled by setting the Statement Cache Size connection string attribute to a value greater than 0.

When statement caching is enabled, however, this property provides a way to selectively add statements to the cache.

Example

```
// C#  
  
using System;  
using System.Data;  
using Oracle.DataAccess.Client;  
  
class AddToStatementCacheSample  
{  
    static void Main()  
    {  
        string constr = "User Id=scott;Password=tiger;Data Source=oracle;" +  
            "statement cache size=10";  
        OracleConnection con = new OracleConnection(constr);  
        con.Open();  
    }  
}
```

```
OracleCommand cmd = new OracleCommand("select * from emp", con);

if (cmd.AddToStatementCache)
    Console.WriteLine("Added to the statement cache:" + cmd.CommandText);
else
    Console.WriteLine("Not added to the statement cache:" + cmd.CommandText);

// The execution of "select * from emp" will be added to the statement cache
// because statement cache size is greater than 0 and OracleCommand's
// AddToStatementCache is true by default.
OracleDataReader readerEmp = cmd.ExecuteReader();

// Do not add "select * from dept" to the statement cache
cmd.CommandText = "select * from dept";
cmd.AddToStatementCache = false;

if (cmd.AddToStatementCache)
    Console.WriteLine("Added to the statement cache:" + cmd.CommandText);
else
    Console.WriteLine("Not added to the statement cache:" + cmd.CommandText);

// The execution of "select * from dept" will not be added to the
// statement cache because AddToStatementCache is set to false.
OracleDataReader readerDept = cmd.ExecuteReader();

// Clean up
con.Dispose();
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)
- ["Statement Caching"](#)
- [ConnectionString](#)

ArrayBindCount

This property specifies if the array binding feature is to be used and also specifies the number of array elements to be bound in the `OracleParameter.Value` property.

Declaration

```
// C#
public int ArrayBindCount {get; set;}
```

Property Value

An `int` value that specifies number of array elements to be bound in the `OracleParameter.Value` property.

Exceptions

ArgumentException - The ArrayBindCount value specified is invalid.

Remarks

Default = 0.

If ArrayBindCount is equal to 0, array binding is not used; otherwise, array binding is used and OracleParameter Value property is interpreted as an array of values. The value of ArrayBindCount must be specified to use the array binding feature.

If neither DbType nor OracleDbType is set, it is strongly recommended that you set ArrayBindCount before setting the OracleParameter Value property so that inference of DbType and OracleDbType from Value can be correctly done.

Array binding is not used by default.

If the XmlCommandType property is set to any value other than None, this property is ignored.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)
- ["Array Binding"](#)
- ["Value "](#)

ArrayBindRowsAffected

This property returns the number of affected rows for each iteration while executing a DML using array binding.

Declaration

```
// C#  
public long[] ArrayBindRowsAffected ;
```

Property Value

A long type

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)
- ["Array Binding"](#)
- ["Value "](#)

BindByName

This property specifies the binding method in the collection.

Declaration

```
// C#  
public bool BindByName {get; set;}
```

Property Value

Returns true if the parameters are bound by name; returns false if the parameters are bound by position.

Remarks

Default = false.

BindByName is ignored under the following conditions:

- The value of the XmlCommandType property is Insert, Update, or Delete.
- The value of the XmlCommandType property is Query, but there are no parameters set on the OracleCommand.

If the XmlCommandType property is OracleXmlCommandType.Query and any parameters are set on the OracleCommand, the BindByName property must be set to true. Otherwise, the following OracleCommand methods throw an InvalidOperationException.

- ExecuteNonQuery
- ExecuteXmlReader
- ExecuteStream
- ExecuteToStream

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)
- ["Array Binding"](#)
- ["Value "](#)

CommandText

This property specifies the SQL statement or stored procedure to run against the Oracle database or the XML data used to store changes to the Oracle database.

Declaration

```
// C#  
public override string CommandText {get; set;}
```

Property Value

A string.

Implements

IDbCommand

Remarks

The default is an empty string.

When the CommandType property is set to StoredProcedure, the CommandText property is set to the name of the stored procedure. The command calls this stored procedure when an Execute method is called.

The effects of XmlCommandType values on CommandText are:

- XmlCommandType = None.
CommandType property determines the contents of CommandText.
- XmlCommandType = Query.
CommandText must be a SQL query. The SQL query should be a select statement. CommandType property is ignored.
- XmlCommandType property is Insert, Update, or Delete.
CommandText must be an XML document. CommandType property is ignored.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

CommandTimeout

This property specifies the minimum number of seconds that the command is allowed to execute before terminating with an exception.

Declaration

```
// C#  
public override int CommandTimeout {get; set;}
```

Property Value

int

Implements

IDbCommand.CommandTimeout

Exceptions

InvalidArgument - The specified value is less than 0.

Remarks

Default is 0 seconds, which enforces no time limit.

When the specified timeout value expires before a command execution finishes, ODP.NET will begin the process of canceling the command. If cancellation is successful, then an exception is thrown with the message ORA-01013: user requested cancel of current operation. Other possible exceptions thrown after a command timeout expiration occurs include ORA-00936 and ORA-00604. If the command is executed in time without any errors, then no exceptions are thrown.

ODP.NET does not guarantee a command will be cancelled at the exact moment the timeout value is reached. The timeout value is the minimum time to allow a command to complete before the cancellation process begins. It is possible for commands to complete after the timeout value expires.

In a situation where multiple OracleCommand objects use the same connection, the timeout expiration on one of the OracleCommand objects may terminate any of the executions on the single connection. To make the timeout expiration of a OracleCommand cancel only its own command execution, simply use one OracleCommand for each connection if that OracleCommand sets the CommandTimeout property to a value greater than 0.

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleCommand Class](#)
- [OracleCommand Members](#)
- <http://msdn.microsoft.com/library> for detailed information about this Microsoft .NET Framework feature

CommandType

This property specifies the command type that indicates how the CommandText property is to be interpreted.

Declaration

```
// C#  
public override CommandType CommandType {get; set;}
```

Property Value

A CommandType.

Exceptions

ArgumentException - The value is not a valid CommandType.

Remarks

Default = CommandType.Text

If the value of the XmlCommandType property is not None, then the CommandType property is ignored. ODP.NET does not support the property value CommandType.TableDirect.

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleCommand Class](#)
- [OracleCommand Members](#)

Connection

This property specifies the OracleConnection object that is used to identify the connection to execute a command.

Declaration

```
// C#  
public OracleConnection Connection {get; set;}
```

Property Value

An OracleConnection object.

Implements

IDbCommand

Remarks

Default = null

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

DesignTimeVisible

This property specifies whether or not the OracleCommand object is visible on designer controls.

Declaration

```
// C#  
public override bool DesignTimeVisible { get; set; }
```

Property Value

A value that indicate whether or not OracleCommand object is visible in a control. The default is true.

Remarks

This property is used by developers to indicate whether or not OracleCommand object is visible in a control.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

FetchSize

This property specifies the size of OracleDataReader's internal cache to store result set data.

Declaration

```
// C#  
public long FetchSize {get; set;}
```

Property Value

A long that specifies the size (in bytes) of the `OracleDataReader`'s internal cache.

Exceptions

`ArgumentException` - The `FetchSize` value specified is invalid.

Remarks

Default = 131072.

The `FetchSize` property is inherited by the `OracleDataReader` that is created by a command execution returning a result set. The `FetchSize` property on the `OracleDataReader` object determines the amount of data the `OracleDataReader` fetches into its internal cache for each database round-trip.

If the `XmlCommandType` property is set to any value other than `None`, this property is ignored.

The `RowSize` and `FetchSize` properties handle UDT and `XMLType` data differently than other scalar data types. Because only a reference to the UDT and `XMLType` data is stored in the ODP.NET's internal cache, the `RowSize` property accounts for only the memory needed for the reference (which is very small) and not the actual size of the UDT and `XMLType` data. Thus, applications can inadvertently fetch a large number of UDT or `XMLType` instances from the database in a single database round-trip. This is because the actual size of UDT and `XMLType` data do not count against the `FetchSize`, and it would require numerous UDT and `XMLType` references to fill up the default cache size of 131072 bytes. Therefore, when fetching UDT or `XMLType` data, the `FetchSize` property must be appropriately configured to control the number of UDT and `XMLType` instances that are to be fetched, rather than the amount of the actual UDT and `XMLType` data to be fetched.

NOTE: For LOB and LONG data types, only the sizes specified in the `InitialLOBFetchSize` and `InitialLONGFetchSize` properties are accounted for by the `RowSize` property in addition to the metadata and reference information that is maintained by the cache for each LOB in the select list.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)
- `OracleDataReader` ["FetchSize"](#)

FetchSqlId

This property indicates whether or not to fetch the `SQL_ID` of the executed statements.

Declaration

```
// C#  
public bool FetchSqlId {get; set;}
```

Property Value

A boolean value

Remarks

By default, the value of this property is false which means SQL_ID would not be fetched for executed statements.

Set this property to true before executing a statement to be able to get the SQL_ID corresponding to the statement being executed.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

ImplicitRefCursors

This property returns an array of OracleRefCursors, where each OracleRefCursor maps to an implicit resultset returned by the stored procedure.

Declaration

```
// C#  
public OracleRefCursor[] ImplicitRefCursors {get; set;}
```

Property Value

An array of OracleRefCursors.

Remarks

This property is populated only when the stored procedure is executed through ExecuteNonQuery and it does not get populated in any other scenarios.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

InitialLOBFetchSize

This property specifies the amount of data that the `OracleDataReader` initially fetches for LOB columns.

Declaration

```
// C#  
public int InitialLOBFetchSize {get; set;}
```

Property Value

An `int` specifying the number of characters or bytes to fetch initially.

Exceptions

`ArgumentException` - The `InitialLOBFetchSize` value specified is invalid.

Remarks

The value of `InitialLOBFetchSize` specifies the initial amount of LOB data that is immediately fetched by the `OracleDataReader`. The property value specifies the number of characters for CLOB and NCLOB data, and the number of bytes for BLOB data.

The `InitialLOBFetchSize` value is used to determine the length of the LOB column data to fetch, if the LOB column is in the select list. If the select list does not contain a LOB column, the `InitialLOBFetchSize` value is ignored.

When `InitialLOBFetchSize` is set to -1, the entire LOB data is prefetched and stored in the fetch array.

Default = 0.

The maximum value supported for `InitialLOBFetchSize` is 2 GB.

`GetOracleBlob` and `GetOracleClob` methods can be used to retrieve any LOBs no matter the `InitialLOBFetchSize` value.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)
- ["Obtaining LOB Data"](#) for more information on setting `InitialLOBFetchSize` values

InitialLONGFetchSize

This property specifies the amount of data that the `OracleDataReader` initially fetches for LONG and LONG RAW columns.

Declaration

```
// C#  
public int InitialLONGFetchSize {get; set;}
```

Property Value

An int specifying the amount.

Exceptions

ArgumentException - The InitialLONGFetchSize value specified is invalid.

Remarks

The maximum value supported for InitialLONGFetchSize is 32767. If this property is set to a higher value, the provider resets it to 32767.

The value of InitialLONGFetchSize specifies the initial amount of LONG or LONG RAW data that is immediately fetched by the OracleDataReader. The property value specifies the number of characters for LONG data and the number of bytes for LONG RAW. To fetch more than the specified InitialLONGFetchSize amount, one of the following must be in the select list:

- Primary key
- ROWID
- Unique columns - (defined as a set of columns on which a unique constraint has been defined or a unique index has been created, where at least one of the columns in the set has a NOT NULL constraint defined on it)

The InitialLONGFetchSize value is used to determine the length of the LONG and LONG RAW column data to fetch if one of the two is in the select list. If the select list does not contain a LONG or a LONG RAW column, the InitialLONGFetchSize value is ignored.

When InitialLONGFetchSize is set to -1, the entire LONG or LONG RAW data is prefetched and stored in the fetch array. Calls to GetString, GetChars, or GetBytes in OracleDataReader allow retrieving the entire data.

Default = 0.

Setting this property to 0 defers the LONG and LONG RAW data retrieval entirely until the application specifically requests it.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)
- ["Obtaining LONG and LONG RAW Data"](#) for further information

Notification

This instance property indicates that there is a notification request for the command.

Declaration

```
// C#  
public OracleNotificationRequest Notification {set; get;}
```

Property Value

A notification request for the command.

Remarks

When a changed notification is first registered, the client listener is started in order to receive any database notification. The listener uses the port number defined in the `OracleDependency.Port` static field. Subsequent change notification registrations use the same listener in the same client process and do not start another listener.

When `Notification` is set to an `OracleNotificationRequest` instance, a notification registration is created (if it has not already been created) when the command is executed. Once the registration is created, the properties of the `OracleNotificationRequest` instance cannot be modified. If the notification registration has already been created, the result set that is associated with the command is added to the existing registration.

When `Notification` is set to null, subsequent command executions do not require a notification request. If a notification request is not required, set the `Notification` property to null, or set the `NotificationAutoEnlist` property to false.

For Continuous Query Notification, a notification request can be used for multiple command executions. In that case, any query result set associated with different commands can be invalidated within the same registration.

When the `OracleDependency.OnChange` event is fired, if the ROWID column is explicitly included in the query (or `AddRowid` property is set to true), then the `Rowid` column contains ROWID values in the `DataTable` referenced by the `OracleNotificationEventArgs.Details` property. This behavior can be overridden by explicitly requesting for an inclusion and exclusion of ROWID values in the `OracleNotificationEventArgs` by setting the `OracleDependency.RowidInfo` to `OracleRowidInfo.Include` or `OracleRowidInfo.Exclude`, respectively.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)
- ["Continuous Query Notification Support"](#)
- [Continuous Query Notification Classes](#)

NotificationAutoEnlist

This instance property indicates whether or not to register for a continuous query notification with the database automatically when the command is executed.

Declaration

```
// C#  
public bool NotificationAutoEnlist {set; get;}
```

Property Value

A bool value indicating whether or not to make a continuous query notification request automatically, when the command is executed. If NotificationAutoEnlist is set to true, and the Notification property is set appropriately, a continuous query notification request is registered automatically; otherwise, no continuous query notification registration is made.

Default value: true

Remarks

A notification request can be used for multiple command executions using the same OracleCommand instance. In that case, set the NotificationAutoEnlist property to true.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)
- ["Continuous Query Notification Support"](#)
- [Continuous Query Notification Classes](#)

Parameters

This property specifies the parameters for the SQL statement or stored procedure.

Declaration

```
// C#  
public OracleParameterCollection Parameters {get;}
```

Property Value

OracleParameterCollection

Implements

IDbCommand

Remarks

Default value = an empty collection

The number of the parameters in the collection must be equal to the number of parameter placeholders within the command text, or an error is raised.

If the command text does not contain any parameter tokens (such as :1, :2), the values in the Parameters property are ignored.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

RowSize

This property specifies the amount of memory needed by the `OracleDataReader` internal cache to store one row of data.

Declaration

```
// C#  
public long RowSize {get;}
```

Property Value

A long that indicates the amount of memory (in bytes) that an `OracleDataReader` needs to store one row of data for the executed query.

Remarks

Default value = 0

The `RowSize` property is set to a nonzero value after the execution of a command that returns a result set. This property can be used at design time or dynamically during runtime, to set the `FetchSize`, based on number of rows. For example, to enable the `OracleDataReader` to fetch `N` rows for each database round-trip, the `OracleDataReader` `FetchSize` property can be set dynamically to `RowSize * N`. Note that for the `FetchSize` to take effect appropriately, it must be set after `OracleCommand.ExecuteReader()` but before `OracleDataReader.Read()`.

ODP.NET now supports values up to 32K for `VARCHAR2`, `NVARCHAR2` or `RAW` type columns in its calculation of `RowSize` value.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)
- `OracleDataReader` ["FetchSize"](#)

RowsToFetchPerRoundTrip

This property specifies the total number of rows to retrieve per database round trip.

Declaration

```
// C#  
public Int64 RowsToFetchPerRoundTrip { get; set; }
```

Property Type

System.Int64

Remarks

This property has no default value. If it is not set, ODP.NET ignores this property.

The row data fetched in a single round trip applies to scalar types only, such as NUMBER and VARCHAR2 columns. If reference data types, such as LOBs, UDTs, and XMLType, exist in the result set, they are retrieved in separate round trips. These reference types can have their own fetch tuning parameters as reference type data sizes can vary in size from row to row, sometimes significantly.

RowsToFetchPerRoundTrip can be set before or after the OracleCommand executes on OracleConnection, OracleCommand, OracleDataReader, or OracleRefCursor. The value can also be changed after initial data fetches so that more or fewer rows are fetched on subsequent round trips. For OracleConfiguration, this property can only be set before the first connection opens.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

SqlId

This property provides the SQL_ID corresponding to the last executed statement on the OracleCommand.

Declaration

```
// C#  
public string SqlId { get; }
```

Property Value

A string value

Remarks

If FetchSqlId was set to false then this property will return null.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

Transaction

This property specifies the `OracleTransaction` object in which the `OracleCommand` executes.

Declaration

```
// C#  
public OracleTransaction Transaction {set; get;}
```

Property Value

`OracleTransaction`

Implements

`IDbCommand`

Remarks

Default value = `null`

`Transaction` returns a reference to the transaction object associated with the `OracleCommand` connection object. Thus the command is executed in whatever transaction context its connection is currently in.

① Note

When this property is accessed through an `IDbCommand` reference, its set accessor method is not operational.

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleCommand Class](#)
- [OracleCommand Members](#)

UpdatedRowSource

This property specifies how query command results are applied to the row to be updated.

Declaration

```
// C#  
public override UpdateRowSource UpdatedRowSource {get; set;}
```

Property Value

An `UpdateRowSource`.

Implements

`IDbCommand`

Exceptions

ArgumentException - The UpdateRowSource value specified is invalid.

Remarks

Always returns UpdateRowSource,

Set accessor throws an ArgumentException if the value is other than UpdateRowSource.None.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

UseEdmMapping

This property Indicates whether or not the OracleCommand object utilizes the Entity Data Model mapping configuration values.

Declaration

```
// C#  
public bool UseEdmMapping
```

Property Value

A bool.

Remarks

Default is false.

The UseEdmMapping property allows user to explicitly specify that the OracleCommand object should use the Entity Data Model mapping configuration values. This enables use of Entity Framework Multiple Result Sets feature.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

XmlCommandType

This property specifies the type of XML operation on the OracleCommand.

Declaration

```
// C#  
public OracleXmlCommandType XmlCommandType {get; set;}
```

Property Value

An OracleXmlCommandType.

Remarks

Default value is None.

XmlCommandType values and usage:

- None - The CommandType property specifies the type of operation.
- Query - CommandText property must be set to a SQL select statement. The query is executed, and the results are returned as an XML document. The SQL select statement in the CommandText and the properties specified by the XmlQueryProperties property are used to perform the operation. The CommandType property is ignored.
- Insert, Update, or Delete - CommandText property is an XML document containing the changes to be made. The XML document in the CommandText and the properties specified by the XmlSaveProperties property are used to perform the operation. The CommandType property is ignored.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

XmlQueryProperties

This property specifies the properties that are used when an XML document is created from the result set of a SQL query statement.

Declaration

```
// C#  
public OracleXmlQueryProperties XmlQueryProperties {get; set;}
```

Property Value

OracleXmlQueryProperties.

Remarks

When a new instance of OracleCommand is created, an instance of OracleXmlQueryProperties is automatically available on the OracleCommand instance through the OracleCommand.XmlQueryProperties property.

A new instance of OracleXmlQueryProperties can be assigned to an OracleCommand instance. Assigning an instance of OracleXmlQueryProperties to the XmlQueryProperties of an

OracleCommand instance creates a new instance of the given OracleXmlQueryProperties instance for the OracleCommand. This way each OracleCommand instance has its own OracleXmlQueryProperties instance.

Use the default constructor to get a new instance of OracleXmlQueryProperties.

Use the OracleXmlQueryProperties.Clone() method to get a copy of an OracleXmlQueryProperties instance.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

XmlSaveProperties

This property specifies the properties that are used when an XML document is used to save changes to the database.

Declaration

```
// C#  
public OracleXmlSaveProperties XmlSaveProperties {get; set;}
```

Property Value

OracleXmlSaveProperties.

Remarks

When a new instance of OracleCommand is created, an instance of OracleXmlSaveProperties is automatically available on the OracleCommand instance through the OracleCommand.XmlSaveProperties property.

A new instance of OracleXmlSaveProperties can be assigned to an OracleCommand instance. Assigning an instance of OracleXmlSaveProperties to the XmlSaveProperties of an OracleCommand instance creates a new instance of the given OracleXmlSaveProperties instance for the OracleCommand. This way each OracleCommand instance has its own OracleXmlSaveProperties instance.

Use the default constructor to get a new instance of OracleXmlSaveProperties.

Use the OracleXmlSaveProperties.Clone() method to get a copy of an OracleXmlSaveProperties instance.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

OracleCommand Public Methods

OracleCommand public methods are listed in [Table 6-16](#).

Table 6-16 OracleCommand Public Methods

Public Method	Description
Cancel	Attempts to cancel a command that is currently executing on a particular connection
Clone	Creates a copy of OracleCommand object
CreateObjRef	Inherited from System.MarshalByRefObject
CreateParameter	Creates a new instance of OracleParameter class
Dispose	Releases any resources or memory allocated by the object
Equals	Inherited from System.Object (Overloaded)
ExecuteNonQuery	Executes a SQL statement or a command using the XmlCommandType and CommandText properties and returns the number of rows affected
ExecuteNonQueryAsync	Returns a Task-based asynchronous version of ExecuteNonQuery() (Overloaded).
ExecuteReader	Executes a command (Overloaded)
ExecuteReaderAsync	Returns a Task-based asynchronous version of OracleCommand.ExecuteReader(), which fetches the result set as an OracleDataReader object. (Overloaded)
ExecuteScalar	Returns the first column of the first row in the result set returned by the query
ExecuteScalarAsync	Returns a Task-based asynchronous version of OracleCommand.ExecuteScalar(), which returns the first column of the first row in the result set returned by the query. (Overloaded)
ExecuteStream	Executes a command using the XmlCommandType and CommandText properties and returns the results in a new Stream object
ExecuteToStream	Executes a command using the XmlCommandType and CommandText properties and appends the results as an XML document to the existing Stream
ExecuteXmlReader	Executes a command using the XmlCommandType and CommandText properties and returns the result as an XML document in a .NET XmlTextReader object
ExecuteXmlReaderAsync	Returns a Task-based asynchronous version of OracleCommand.ExecuteXmlReader(), which fetches the result set as an XmlReader object (Overloaded)
GetHashCode	Inherited from System.Object
GetLifetimeService	Inherited from System.MarshalByRefObject
GetType	Inherited from System.Object
InitializeLifetimeService	Inherited from System.MarshalByRefObject

Table 6-16 (Cont.) OracleCommand Public Methods

Public Method	Description
Prepare	<i>This method is a no-op</i>
ToString	Inherited from <code>System.Object</code>

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleCommand Class](#)
- [OracleCommand Members](#)

Cancel

This method attempts to cancel a command that is currently executing on a particular connection.

Declaration

```
// C#  
public override void Cancel();
```

Implements

```
IDbCommand.Cancel
```

Remarks

If cancellation of the command succeeds, an exception is thrown. If cancellation is not successful, no exception is thrown. If there is no command being executed at the time of the `Cancel` invocation, `Cancel` does nothing. Invoking the `Cancel` method does not guarantee that the command executing at the time will always be cancelled. The execution may complete before it can be terminated. In such cases, no exception is thrown.

Commands cancellation will stop the command execution stage or results fetching stage if it is a query. If these stages have completed, `Cancel` does nothing.

ODP.NET sends a cancellation message to the database server when the `Cancel` method is invoked. There may be a delay when cancelling a running PL/SQL program. This delay is specific to PL/SQL's design in how it handles command cancellations.

When managed ODP.NET or ODP.NET Core invokes `Cancel`, it is deterministic. That means the command executed by the `OracleCommand` object is the only command that could be canceled. Unmanaged ODP.NET `Cancel` is non-deterministic.

Non-determinism in unmanaged ODP.NET means that when multiple `OracleCommand` objects share the same connection, only one command can be executed on that connection at any one time. When it is invoked, the `Cancel` method attempts to cancel the statement currently running on the connection that the `OracleCommand` object is using to execute the command. However, when multiple `OracleCommand` objects execute statements on the same connection simultaneously, issuing a `Cancel` method invocation may cancel any of the issued commands. This is because the command designated for cancellation may complete before the `Cancel`

invocation is effective. If this happens, a command executed by a different `OracleCommand` could be cancelled instead.

There are several ways to avoid this non-deterministic situation that the `Cancel` method can cause:

- The application can create just one `OracleCommand` object for each connection. Doing so assures that the `Cancel` invocation only cancels commands executed by the `OracleCommand` object using a particular connection.
- Command executions in the application are synchronized between `OracleCommand` objects that use the same connection.

These suggestions do not apply if `Cancel` is not used in the application.

Because the termination on the currently running execution is non-deterministic, it is recommended that any *non-atomic* SQL or PL/SQL execution be started within a transaction. When the command execution successfully terminates with an exception of ORA-01013: user requested cancel of current operation, the transaction can be rolled back for data integrity. Other possible exceptions thrown after a command cancellation occurs include ORA-00936 and ORA-00604. Examples of non-atomic execution are collections of DML command executions that are executed one-by-one and multiple DML commands that are part of a PL/SQL stored procedure or function.

Example

```
// C#

// This example shows how command executions can be cancelled in a
// deterministic way even if multiple commands are executed on a single
// connection. This is accomplished by synchronizing threads through events.
// Since the Cancel method terminates the currently running operation on the
// connection, threads must be serialized if multiple threads are using the
// same connection to execute server round-trip incurring operations.
// Furthermore, the example shows how the execution and cancel threads should
// be synchronized so that nth iteration of the command execution does not
// inappropriately cancel the (n+1)th command executed by the same thread.

using System;
using System.Data;
using Oracle.DataAccess.Client;
using System.Threading;

class CancelSample
{
    private OracleCommand cmd;
    Thread t1, t2;
    // threads signal following events when assigned operations are completed

    private AutoResetEvent ExecuteEvent = new AutoResetEvent(false);
    private AutoResetEvent CancelEvent = new AutoResetEvent(false);
    private AutoResetEvent FinishedEvent = new AutoResetEvent(false);
    AutoResetEvent[] ExecuteAndCancel = new AutoResetEvent[2];

    // Default constructor
    CancelSample()
    {
        cmd = new OracleCommand("select * from all_objects",
            new OracleConnection("user id=scott;password=tiger;data source=oracle"));
        ExecuteAndCancel[0] = ExecuteEvent;
        ExecuteAndCancel[1] = CancelEvent;
    }
}
```

```

// Constructor that takes a particular command and connection
CancelSample(string command, OracleConnection con)
{
    cmd = new OracleCommand(command, con);
    ExecuteAndCancel[0] = ExecuteEvent;
    ExecuteAndCancel[1] = CancelEvent;
}

// Execution of the command
public void Execute()
{
    OracleDataReader reader = null;
    try
    {
        Console.WriteLine("Execute.");
        reader = cmd.ExecuteReader();
        Console.WriteLine("Execute Done.");
        reader.Close();
    }
    catch(Exception e)
    {
        Console.WriteLine("The command has been cancelled.", e.Message);
    }
    Console.WriteLine("ExecuteEvent.Set()");
    ExecuteEvent.Set();
}

// Canceling of the command
public void Cancel()
{
    try
    {
        // cancel query if it takes longer than 100 ms to finish execution
        System.Threading.Thread.Sleep(100);
        Console.WriteLine("Cancel.");
        cmd.Cancel();
    }
    catch (Exception e)
    {
        Console.WriteLine(e.ToString());
    }
    Console.WriteLine("Cancel done.");
    Console.WriteLine("CancelEvent.Set()");
    CancelEvent.Set();
}

// Execution of the command with a potential of cancelling
public void ExecuteWithinLimitedTime()
{
    for (int i = 0; i < 5; i++)
    {
        Monitor.Enter(typeof(CancelSample));
        try
        {
            Console.WriteLine("Executing " + this.cmd.CommandText);
            ExecuteEvent.Reset();
            CancelEvent.Reset();
            t1 = new Thread(new ThreadStart(this.Execute));
            t2 = new Thread(new ThreadStart(this.Cancel));
            t1.Start();
            t2.Start();

```

```

    }
    finally
    {
        WaitHandle.WaitAll(ExecuteAndCancel);
        Monitor.Exit(typeof(CancelSample));
    }
}
FinishedEvent.Set();
}
[MTAThread]
static void Main()
{
    try
    {
        AutoResetEvent[] ExecutionCompleteEvents = new AutoResetEvent[3];

        // Create the connection that is to be used by three commands
        OracleConnection con = new OracleConnection("user id=scott;" +
            "password=tiger;data source=oracle");
        con.Open();

        // Create instances of CancelSample class
        CancelSample test1 = new CancelSample("select * from all_objects", con);
        CancelSample test2 = new CancelSample("select * from all_objects, emp",
            con);
        CancelSample test3 = new CancelSample("select * from all_objects, dept",
            con);

        // Create threads for each CancelSample object instance
        Thread t1 = new Thread(new ThreadStart(test1.ExecuteWithinLimitedTime));
        Thread t2 = new Thread(new ThreadStart(test2.ExecuteWithinLimitedTime));
        Thread t3 = new Thread(new ThreadStart(test3.ExecuteWithinLimitedTime));

        // Obtain a handle to an event from each object
        ExecutionCompleteEvents[0] = test1.FinishedEvent;
        ExecutionCompleteEvents[1] = test2.FinishedEvent;
        ExecutionCompleteEvents[2] = test3.FinishedEvent;

        // Start all threads to execute three commands using a single connection
        t1.Start();
        t2.Start();
        t3.Start();

        // Wait for all three commands to finish executing/canceling before
        //closing the connection
        WaitHandle.WaitAll(ExecutionCompleteEvents);
        con.Close();
    }
    catch (Exception e)
    {
        Console.WriteLine(e.ToString());
    }
}
}

```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)
- <http://msdn.microsoft.com/library> for detailed information about this Microsoft .NET Framework feature

Clone

This method creates a copy of an `OracleCommand` object.

Declaration

```
// C#  
public object Clone();
```

Return Value

An `OracleCommand` object.

Implements

`ICloneable`

Remarks

The cloned object has the same property values as that of the object being cloned.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

CreateParameter

This method creates a new instance of `OracleParameter` class.

Declaration

```
// C#  
public OracleParameter CreateParameter();
```

Return Value

A new `OracleParameter` with default values.

Implements

IDbCommand

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

Dispose

This method releases any resources or memory allocated by the object.

Declaration

```
// C#  
public void Dispose();
```

Implements

IDisposable

Remarks

The Dispose method also closes the OracleCommand object.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

ExecuteNonQuery

This method executes a SQL statement or a command using the XmlCommandType and CommandText properties and returns the number of rows affected.

Declaration

```
// C#  
public override int ExecuteNonQuery();
```

Return Value

The number of rows affected.

Implements

IDbCommand

Exceptions

InvalidOperationException - The command cannot be executed.

Remarks

ExecuteNonQuery returns the number of rows affected, for the following:

- If the command is UPDATE, INSERT, or DELETE and the XmlCommandType property is set to OracleXmlCommandType.None.
- If the XmlCommandType property is set to OracleXmlCommandType.Insert, OracleXmlCommandType.Update, OracleXmlCommandType.Delete.

For all other types of statements, the return value is -1.

ExecuteNonQuery is used for either of the following:

- Catalog operations (for example, querying the structure of a database or creating database objects such as tables).
- Changing the data in a database without using a DataSet, by executing UPDATE, INSERT, or DELETE statements.
- Changing the data in a database using an XML document.

Although ExecuteNonQuery does not return any rows, it populates any output parameters or return values mapped to parameters with data.

If the XmlCommandType property is set to OracleXmlCommandType.Query then ExecuteNonQuery executes the select statement in the CommandText property, and if successful, returns -1. The XML document that is generated is discarded. This is useful for determining if the operation completes successfully without getting the XML document back as a result.

If the XmlCommandType property is set to OracleXmlCommandType.Insert, OracleXmlCommandType.Update, or OracleXmlCommandType.Delete, then the value of the CommandText property is an XML document. ExecuteNonQuery saves the changes in that XML document to the table or view that is specified in the XmlSaveProperties property. The return value is the number of rows that are processed in the XML document. Also, each row in the XML document could affect multiple rows in the database, but the return value is still the number of rows in the XML document.

Example

```
// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;

class ExecuteNonQuerySample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleCommand cmd = new OracleCommand(
            "select sal from emp where empno=7934", con);

        object sal = cmd.ExecuteScalar();
    }
}
```

```
Console.WriteLine("Employee sal before update: " + sal);

cmd.CommandText = "update emp set sal = sal + .01 where empno=7934";

// Auto-commit changes
int rowsUpdated = cmd.ExecuteNonQuery();

if (rowsUpdated > 0)
{
    cmd.CommandText = "select sal from emp where empno=7934";
    sal = cmd.ExecuteScalar();
    Console.WriteLine("Employee sal after update: " + sal);
}

// Clean up
cmd.Dispose();
con.Dispose();
}
}
```

Requirements

For XML support, this method requires Oracle9i XML Developer's Kits (Oracle XDK) or later, to be installed in the database. Oracle XDK can be downloaded from Oracle Technology Network (OTN).

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)
- <http://www.oracle.com/technetwork/index.html>

ExecuteNonQueryAsync

Overload List:

ExecuteNonQueryAsync returns a Task-based asynchronous version of ExecuteNonQuery().

- [ExecuteNonQueryAsync\(\)](#)

This method returns an asynchronous version of ExecuteNonQuery(), which executes the command against its connection object and returns the number of rows affected.

- [ExecuteNonQueryAsync\(Cancellation_token cancellationToken\)](#)

This method returns an asynchronous version of ExecuteNonQuery(), which executes the command against its connection object and returns the number of rows affected with a Cancellation_token.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

ExecuteNonQueryAsync()

This method returns an asynchronous version of `ExecuteNonQuery()`, which executes the command against its connection object and returns the number of rows affected.

Declaration

```
// C#  
public Task<int> int ExecuteNonQueryAsync();
```

Return Value

A Task object.

Implements

DbCommand

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

ExecuteNonQueryAsync(CancellationToken cancellationToken)

This method returns an asynchronous version of `ExecuteNonQuery()`, which executes the command against its connection object and returns the number of rows affected with a `CancellationToken`.

Declaration

```
// C#  
public override Task<int> int ExecuteNonQueryAsync(CancellationToken cancellationToken);
```

Return Value

A Task object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Implements

DbCommand

Exceptions

InvalidOperationException - The command cannot be executed.

Example

```
using Oracle.ManagedDataAccess.Client;
using System;
using System.Threading;
using System.Threading.Tasks;

namespace AsyncApp
{
    class AsyncDemo
    {
        static async Task Main()
        {
            string connection = "User Id=HR; Password=<PASSWORD>; Data Source=oracle;";
            OracleConnection oc = new OracleConnection(connection);
            await oc.OpenAsync(CancellationToken.None);
            Console.WriteLine("Connection opened successfully");

            OracleCommand cmd = oc.CreateCommand();
            cmd.CommandText = "insert into tab1 values(1)";
            Task task = cmd.ExecuteNonQueryAsync(CancellationToken.None);
            await task;
            Console.WriteLine("Query executed");
            oc.Close();
        }
    }
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

ExecuteReader

Overload List:

ExecuteReader executes a command specified in the CommandText.

- [ExecuteReader\(\)](#)
This method executes a command specified in the CommandText and returns an OracleDataReader object.
- [ExecuteReader\(CommandBehavior\)](#)
This method executes a command specified in the CommandText and returns an OracleDataReader object, using the specified CommandBehavior value.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

ExecuteReader()

This method executes a command specified in the CommandText and returns an OracleDataReader object.

Declaration

```
// C#  
public OracleDataReader ExecuteReader();
```

Return Value

An OracleDataReader.

Implements

IDbCommand

Exceptions

InvalidOperationException - The command cannot be executed.

Remarks

When the CommandType property is set to CommandType.StoredProcedure, the CommandText property should be set to the name of the stored procedure.

The specified command executes this stored procedure when ExecuteReader is called. If parameters for the stored procedure consist of REF CURSOR objects, behavior differs depending on whether ExecuteReader() or ExecuteNonQuery() is called. If ExecuteReader() is invoked, REF CURSOR objects can be accessed through the OracleDataReader that is returned. If more than one REF CURSOR is returned from a single execution, subsequent REF CURSOR objects can be accessed sequentially by the NextResult method on the OracleDataReader. If the ExecuteNonQuery method is invoked, the output parameter value can be cast to a OracleRefCursor type and the OracleRefCursor object then can be used to either populate a DataSet or create an OracleDataReader object from it. This approach provides random access to all the REF CURSOR objects returned as output parameters.

The value of 100 is used for the FetchSize. If 0 is specified, no rows are fetched. For further information, see ["Obtaining LONG and LONG RAW Data"](#).

If the value of the XmlCommandType property is set to OracleXmlCommandType.Insert, OracleXmlCommandType.Update, OracleXmlCommandType.Delete, or OracleXmlCommandType.Query then the ExecuteReader method throws an InvalidOperationException.

Example

```
// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;

class ExecuteReaderSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleCommand cmd = new OracleCommand("select ename from emp", con);

        OracleDataReader reader = cmd.ExecuteReader();

        while (reader.Read())
        {
            Console.WriteLine("Employee Name : " + reader.GetString(0));
        }

        // Clean up
        reader.Dispose();
        cmd.Dispose();
        con.Dispose();
    }
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)
- ["OracleRefCursor Class"](#)

ExecuteReader(CommandBehavior)

This method executes a command specified in the CommandText and returns an OracleDataReader object, using the specified behavior.

Declaration

```
// C#
public OracleDataReader ExecuteReader(CommandBehavior behavior);
```

Parameters

- *behavior*
The expected behavior.

Return Value

An `OracleDataReader`.

Implements

`IDbCommand`

Exceptions

`InvalidOperationException` - The command cannot be executed.

Remarks

A description of the results and the effect on the database of the query command is indicated by the supplied *behavior* that specifies command behavior.

For valid `CommandBehavior` values and for the command behavior of each `CommandBehavior` enumerated type, read the .NET Framework documentation.

When the `CommandType` property is set to `CommandType.StoredProcedure`, the `CommandText` property should be set to the name of the stored procedure. The command executes this stored procedure when `ExecuteReader()` is called.

If the stored procedure returns stored REF CURSORS, read the section on `OracleRefCursors` for more details. See "[OracleRefCursor Class](#)".

The value of 100 is used for the `FetchSize`. If 0 is specified, no rows are fetched. For more information, see "[Obtaining LONG and LONG RAW Data](#)".

If the value of the `XmlCommandType` property is set to `OracleXmlCommandType.Insert`, `OracleXmlCommandType.Update`, `OracleXmlCommandType.Delete`, or `OracleXmlCommandType.Query` then the `ExecuteReader` method throws an `InvalidOperationException`.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleCommand Class](#)
- [OracleCommand Members](#)
- "[OracleRefCursor Class](#)"

ExecuteReaderAsync

Overload List:

This method returns a Task-based asynchronous version of `OracleCommand.ExecuteReader()`, which fetches the result set as an `OracleDataReader` object.

- [ExecuteReaderAsync\(\)](#)

This method returns a Task-based asynchronous version of `OracleCommand.ExecuteReader()`, which fetches the result set as an `OracleDataReader` object.

- [ExecuteReaderAsync\(CancellationToken cancellationToken\)](#)
This method returns a Task-based asynchronous version of `OracleCommand.ExecuteReader()`, which fetches the result set as an `OracleDataReader` object.
- [ExecuteReaderAsync\(CommandBehavior behavior\)](#)
This method returns a Task-based asynchronous version of `OracleCommand.ExecuteReader()`, which fetches the result set as an `OracleDataReader` object.
- [ExecuteReaderAsync\(CommandBehavior behavior, CancellationToken cancellationToken\)](#)
This method returns a Task-based asynchronous version of `OracleCommand.ExecuteReader()`, which fetches the result set as an `OracleDataReader` object.

Example (Using all overloads)

```
using Oracle.ManagedDataAccess.Client;
using System;
using System.Data;
using System.Threading;

namespace AsyncApp
{
    class AsyncDemo
    {
        static async Task Main()
        {
            string connectionString = "User Id=HR; Password=<PASSWORD>; Data Source=oracle;";
            OracleConnection oc = new OracleConnection(connectionString);
            await oc.OpenAsync(CancellationToken.None);

            OracleCommand cmd = oc.CreateCommand();
            cmd.CommandText = "select * from demo_table";

            OracleDataReader reader;

            reader = await cmd.ExecuteReaderAsync();
            while (await reader.ReadAsync(CancellationToken.None))
            {
                Console.WriteLine(reader.GetValue(0));
            }

            reader = await cmd.ExecuteReaderAsync(CancellationToken.None);
            while (await reader.ReadAsync(CancellationToken.None))
            {
                Console.WriteLine(reader.GetValue(0));
            }

            reader = await cmd.ExecuteReaderAsync(CommandBehavior.Default);
            while (await reader.ReadAsync(CancellationToken.None))
            {
                Console.WriteLine(reader.GetValue(0));
            }

            reader = await cmd.ExecuteReaderAsync(CommandBehavior.Default,
            CancellationToken.None);
            while (await reader.ReadAsync(CancellationToken.None))
            {
```



```
        Console.WriteLine(reader.GetValue(0));  
    }  
} }  
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

ExecuteReaderAsync()

This method returns a Task-based asynchronous version of `OracleCommand.ExecuteReader()`, which fetches the result set as an `OracleDataReader` object.

Declaration

```
// C#  
public Task<OracleDataReader> ExecuteReaderAsync();
```

Return Value

`Task<OracleDataReader>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Implements

`DbCommand`

Exceptions

`InvalidOperationException` - The command cannot be executed.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

ExecuteReaderAsync(CancellationToken cancellationToken)

This method returns a Task-based asynchronous version of `OracleCommand.ExecuteReader()`, which fetches the result set as an `OracleDataReader` object.

Declaration

```
// C#  
public Task<OracleDataReader> ExecuteReaderAsync(CancellationToken cancellationToken);
```

Parameters

cancellationToken - The input cancellation token which can be used by the application to cancel the task before command timeout occurs.

Return Value

Task<OracleDataReader> object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Implements

DbCommand

Exceptions

InvalidOperationException - The command cannot be executed.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

ExecuteReaderAsync(CommandBehavior behavior)

This method returns a Task-based asynchronous version of OracleCommand.ExecuteReader(), which fetches the result set as an OracleDataReader object.

Declaration

```
// C#  
public Task<OracleDataReader> ExecuteReaderAsync(CommandBehavior behavior);
```

Parameters

behavior - the expected behavior. Provides a description of the results of the query and its effect on the database.

Return Value

Task<OracleDataReader> object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Implements

DbCommand

Exceptions

InvalidOperationException - The command cannot be executed.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

ExecuteReaderAsync(CommandBehavior behavior, CancellationToken cancellationToken)

This method returns a Task-based asynchronous version of `OracleCommand.ExecuteReader()`, which fetches the result set as an `OracleDataReader` object.

Declaration

```
// C#  
public Task<OracleDataReader> ExecuteReaderAsync(CommandBehavior behavior,  
CancellationToken cancellationToken);
```

Parameters

- `behavior` - the expected behavior. Provides a description of the results of the query and its effect on the database.
- `cancellationToken` - the input cancellation token which can be used by the application to cancel the task before command timeout occurs.

Return Value

`Task<OracleDataReader>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Implements

`DbCommand`

Exceptions

`InvalidOperationException` - The command cannot be executed.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

ExecuteScalar

This method executes the query using the connection, and returns the first column of the first row in the result set returned by the query.

Declaration

```
// C#  
public override object ExecuteScalar();
```

Return Value

An object which represents the value of the first row, first column.

Implements

IDbCommand

Exceptions

InvalidOperationException - The command cannot be executed.

Remarks

Extra columns or rows are ignored. ExecuteScalar retrieves a single value (for example, an aggregate value) from a database. This requires less code than using the ExecuteReader() method, and then performing the operations necessary to generate the single value using the data returned by an OracleDataReader.

If the query does not return any row, it returns null.

The ExecuteScalar method throws an InvalidOperationException, if the value of the XmlCommandType property is set to one of the following OracleXmlCommandType values: Insert, Update, Delete, Query.

Example

```
// C#  
  
using System;  
using System.Data;  
using Oracle.DataAccess.Client;  
  
class ExecuteScalarSample  
{  
    static void Main()  
    {  
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";  
        OracleConnection con = new OracleConnection(constr);  
        con.Open();  
  
        OracleCommand cmd = new OracleCommand("select count(*) from emp", con);  
  
        object count = cmd.ExecuteScalar();  
  
        Console.WriteLine("There are {0} rows in table emp", count);  
  
        // Clean up  
        cmd.Dispose();  
        con.Dispose();  
    }  
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

ExecuteStream

This method executes a command using the `XmlCommandType` and `CommandText` properties and returns the result as an XML document in a new `Stream` object.

Declaration

```
// C#  
public Stream ExecuteStream();
```

Return Value

A `Stream`.

Remarks

The behavior of `ExecuteStream` varies depending on the `XmlCommandType` property value:

- `XmlCommandType = OracleXmlCommandType.None`
`ExecuteStream` throws an `InvalidOperationException`.
- `XmlCommandType = OracleXmlCommandType.Query`
`ExecuteStream` executes the select statement in the `CommandText` property, and if successful, returns an `OracleClob` object containing the XML document that was generated. `OracleClob` contains Unicode characters.

If the SQL query does not return any rows, then `ExecuteStream` returns an `OracleClob` object containing an empty XML document.
- `XmlCommandType = OracleXmlCommandType.Insert, OracleXmlCommandType.Update, or OracleXmlCommandType.Delete`.

The value of the `CommandText` property is an XML document. `ExecuteStream` saves the data in that XML document to the table or view that is specified in the `XmlSaveProperties` property and an empty `OracleClob` is returned.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)
- *Oracle XML DB Developer's Guide*
- <http://www.oracle.com/technetwork/index.html>

ExecuteToStream

This method executes a command using the `XmlCommandType` and `CommandText` properties and appends the result as an XML document to the existing `Stream` provided by the application.

Declaration

```
// C#  
public void ExecuteToStream(Stream outputStream);
```

Parameters

- `outputStream`
A `Stream`.

Remarks

The behavior of `ExecuteToStream` varies depending on the `XmlCommandType` property value:

- `XmlCommandType = OracleXmlCommandType.None`
`ExecuteToStream` throws an `InvalidOperationException`.
- `XmlCommandType = OracleXmlCommandType.Query`
`ExecuteToStream` executes the select statement in the `CommandText` property, and if successful, appends the XML document that was generated to the given `Stream`.
If the SQL query does not return any rows, then nothing is appended to the given `Stream`. The character set of the appended data is Unicode.
- `XmlCommandType = OracleXmlCommandType.Insert`, `OracleXmlCommandType.Update`, or `OracleXmlCommandType.Delete`
The value of the `CommandText` property is an XML document. `ExecuteToStream` saves the changes in that XML document to the table or view that is specified in the `XmlSaveProperties` property. Nothing is appended to the given `Stream`.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleCommand Class](#)
- [OracleCommand Members](#)
- *Oracle XML DB Developer's Guide*
- <http://www.oracle.com/technetwork/index.html>

ExecuteScalarAsync

Overload List:

This method returns a Task-based asynchronous version of `OracleCommand.ExecuteScalar()`, which returns the first column of the first row in the result set returned by the query.

- [ExecuteScalarAsync\(\)](#)

This method returns a Task-based asynchronous version of `OracleCommand.ExecuteScalar()`, which returns the first column of the first row in the result set returned by the query.

- [ExecuteScalarAsync\(CancellationToken cancellationToken\)](#)

This method returns a Task-based asynchronous version of `OracleCommand.ExecuteScalar()` with input `CancellationToken`, which returns the first column of the first row in the result set returned by the query.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

ExecuteScalarAsync()

This method returns a Task-based asynchronous version of `OracleCommand.ExecuteScalar()`, which returns the first column of the first row in the result set returned by the query.

Declaration

```
// C#  
public Task<object> ExecuteScalarAsync();
```

Return Value

`Task<object>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Implements

`DbCommand`

Exceptions

`InvalidOperationException` - The command cannot be executed.

Remarks

This will call into the `ExecuteScalarAsync` implementation with argument `'cancellationToken'` passed as `CancellationToken.None`.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

ExecuteScalarAsync(CancellationToken cancellationToken)

This method returns a Task-based asynchronous version of `OracleCommand.ExecuteScalar()`, which returns the first column of the first row in the result set returned by the query.

Declaration

```
// C#  
public override Task<object> ExecuteScalarAsync(CancellationToken cancellationToken);
```

Parameters

`cancellationToken` - The input cancellation token which can be used by the application to cancel the task before command timeout occurs.

Return Value

`Task<object>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Implements

`DbCommand`

Exceptions

`InvalidOperationException` - The command cannot be executed.

Example

```
using Oracle.ManagedDataAccess.Client;  
using System.Threading;  
  
namespace AsyncApp  
{  
    class AsyncDemo  
    {  
        static async Task Main()  
        {  
            string connectionString = "User Id=HR; Password=<PASSWORD>; Data Source=oracle;";  
            OracleConnection oc = new OracleConnection(connectionString);  
            await oc.OpenAsync(CancellationToken.None);  
  
            OracleCommand cmd = oc.CreateCommand();  
            cmd.CommandText = "select * from demo_table";  
  
            var value = await cmd.ExecuteScalarAsync(CancellationToken.None);  
        }  
    }  
}
```


See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

ExecuteXmlReader

This method executes the command using the `XmlCommandType` and `CommandText` properties and returns the result as an XML document in a .NET `XmlTextReader` object.

Declaration

```
// C#  
public XmlReader ExecuteXmlReader();
```

Return Value

An `XmlReader`.

Remarks

The behavior of `ExecuteXmlReader` varies depending on the `XmlCommandType` property value:

- `XmlCommandType = OracleXmlCommandType.None`
`ExecuteStream` and `ExecuteToStream` throw an `InvalidOperationException`.
- `XmlCommandType = OracleXmlCommandType.Query`
`ExecuteXmlReader` executes the select statement in the `CommandText` property, and if successful, returns a .NET `XmlTextReader` object containing the XML document that was generated.

If the XML document is empty, which can happen if the SQL query does not return any rows, then an empty .NET `XmlTextReader` object is returned.
- `XmlCommandType = OracleXmlCommandType.Insert`, `OracleXmlCommandType.Update`, or `OracleXmlCommandType.Delete`.

The value of the `CommandText` property is an XML document, and `ExecuteXmlReader` saves the changes in that XML document to the table or view that is specified in the `XmlSaveProperties` property. An empty .NET `XmlTextReader` object is returned.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)
- *Oracle XML DB Developer's Guide*
- <http://www.oracle.com/technetwork/index.html>

ExecuteXmlReaderAsync

Overload List:

The method `ExecuteXmlReaderAsync` returns a Task-based asynchronous version of `OracleCommand.ExecuteXmlReader()`, which fetches the result set as an `XmlReader` object.

- [ExecuteXmlReaderAsync\(\)](#)

This method returns a Task-based asynchronous version of `OracleCommand.ExecuteXmlReader()`, which fetches the result set as an `XmlReader` object.

- [ExecuteXmlReaderAsync\(CancellationTokentoken\)](#)

This method returns a Task-based asynchronous version of `OracleCommand.ExecuteXmlReader()`, which fetches the result set as an `XmlReader` object.

Example (includes overloads)

```
using Oracle.ManagedDataAccess.Client;
using System;
using System.Threading;
using System.Xml;

namespace AsyncApp
{
    class AsyncDemo
    {
        static async Task Main()
        {
            string connectionString = "User Id=HR; Password=<PASSWORD>; Data Source=oracle;";
            OracleConnection oc = new OracleConnection(connectionString);
            await oc.OpenAsync(CancellationToken.None);

            OracleCommand cmd = oc.CreateCommand();
            cmd.CommandText = "select * from demo_table";

            XmlReader reader;

            reader = await cmd.ExecuteXmlReaderAsync();
            while (reader.Read())
            {
                Console.WriteLine(reader.Value);
            }

            reader = await cmd.ExecuteXmlReaderAsync(CancellationToken.None);
            while (reader.Read())
            {
                Console.WriteLine(reader.Value);
            }
        }
    }
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

ExecuteXmlReaderAsync()

This method returns a Task-based asynchronous version of `OracleCommand.ExecuteXmlReader()`, which fetches the result set as an `XmlReader` object.

Declaration

```
// C#  
public Task<XmlReader> ExecuteXmlReaderAsync();
```

Return Value

`Task<XmlReader>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Implements

`DbCommand`

Exceptions

`InvalidOperationException` - The command cannot be executed.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

ExecuteXmlReaderAsync(CancellationTokens)

This method returns a Task-based asynchronous version of `OracleCommand.ExecuteXmlReader()`, which fetches the result set as an `XmlReader` object.

Declaration

```
// C#  
public Task<XmlReader> ExecuteXmlReaderAsync(CancellationToken cancellationToken);
```

Parameter

cancellationToken - The input cancellation token which can be used by the application to cancel the task before command timeout occurs.

Return Value

Task<XmlReader> object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Implements

DbCommand

Exceptions

InvalidOperationException - The command cannot be executed.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

Prepare

This method is not supported.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

OracleCommandBuilder Class

An OracleCommandBuilder object provides automatic SQL generation for the OracleDataAdapter when updates are made to the database.

Class Inheritance

System.Object

System.MarshalByRefObject

System.ComponentModel.Component

System.Data.Common.DbCommandBuilder

Oracle.DataAccess.Client.OracleCommandBuilder

Declaration

```
// C#  
public sealed class OracleCommandBuilder : DbCommandBuilder
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

OracleCommandBuilder automatically generates SQL statements for single-table updates when the SelectCommand property of the OracleDataAdapter is set. An exception is thrown if the DataSet contains multiple tables. The OracleCommandBuilder registers itself as a listener for RowUpdating events whenever its DataAdapter property is set. Only one OracleDataAdapter object and one OracleCommandBuilder object can be associated with each other at one time.

To generate INSERT, UPDATE, or DELETE statements, the OracleCommandBuilder uses ExtendedProperties within the DataSet to retrieve a required set of metadata. If the SelectCommand is changed after the metadata is retrieved (for example, after the first update), the RefreshSchema method should be called to update the metadata.

OracleCommandBuilder first looks for the metadata from the ExtendedProperties of the DataSet; if the metadata is not available, OracleCommandBuilder uses the SelectCommand property of the OracleDataAdapter to retrieve the metadata.

Example

The following example performs an update on the EMP table. It uses the OracleCommandBuilder object to create the UpdateCommand for the OracleDataAdapter object when OracleDataAdapter.Update() is called.

```
// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;

class OracleCommandBuilderSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        string cmdstr = "SELECT empno, sal from emp";

        // Create the adapter with the selectCommand txt and the
        // connection string
        OracleDataAdapter adapter = new OracleDataAdapter(cmdstr, constr);
```

```
// Create the builder for the adapter to automatically generate
// the Command when needed
OracleCommandBuilder builder = new OracleCommandBuilder(adapter);

// Create and fill the DataSet using the EMP
DataSet dataset = new DataSet();
adapter.Fill(dataset, "EMP");

// Get the EMP table from the dataset
DataTable table = dataset.Tables["EMP"];

// Indicate DataColumn EMPNO is unique
// This is required by the OracleCommandBuilder to update the EMP table
table.Columns["EMPNO"].Unique = true;

// Get the first row from the EMP table
DataRow row = table.Rows[0];

// Update the salary
double sal = double.Parse(row["SAL"].ToString());
row["SAL"] = sal + .01;

// Now update the EMP using the adapter
// The OracleCommandBuilder will create the UpdateCommand for the
// adapter to update the EMP table
adapter.Update(dataset, "EMP");

Console.WriteLine("Row updated successfully");
}
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Members](#)
- [OracleCommandBuilder Constructors](#)
- [OracleCommandBuilder Static Methods](#)
- [OracleCommandBuilder Properties](#)
- [OracleCommandBuilder Public Methods](#)
- [OracleCommandBuilder Events](#)

OracleCommandBuilder Members

OracleCommandBuilder members are listed in the following tables.

OracleCommandBuilder Constructors

OracleCommandBuilder constructors are listed in [Table 6-17](#).

Table 6-17 OracleCommandBuilder Constructors

Constructor	Description
OracleCommandBuilder Constructors	Instantiates a new instance of OracleCommandBuilder class (Overloaded)

OracleCommandBuilder Static Methods

OracleCommandBuilder static methods are listed in [Table 6-18](#).

Table 6-18 OracleCommandBuilder Static Methods

Method	Description
DeriveParameters	Queries for the parameters of a stored procedure or function, represented by a specified OracleCommand, and populates the OracleParameterCollection of the command with the return values
Equals	Inherited from System.Object (Overloaded)

OracleCommandBuilder Properties

OracleCommandBuilder properties are listed in [Table 6-19](#).

Table 6-19 OracleCommandBuilder Properties

Property	Description
Container	Inherited from System.ComponentModel.Component
CaseSensitive	Indicates whether or not double quotes are used around Oracle object names when generating SQL statements
CatalogLocation	<i>Not Supported</i>
CatalogSeparator	<i>Not Supported</i>
ConflictOption	<i>Not Supported</i>
DataAdapter	Indicates the OracleDataAdapter for which the SQL statements are generated
QuotePrefix	Specifies the beginning character or characters used to specify database objects whose names contain special characters such as spaces or reserved words
QuoteSuffix	Specifies the ending character or characters used to specify database objects whose names contain special characters such as spaces or reserved words
SchemaSeparator	Specifies the character to be used for the separator between the schema identifier and other identifiers
Site	Inherited from System.ComponentModel.Component

OracleCommandBuilder Public Methods

OracleCommandBuilder public methods are listed in [Table 6-20](#).

Table 6-20 OracleCommandBuilder Public Methods

Public Method	Description
CreateObjRef	Inherited from System.MarshalByRefObject
Dispose	Inherited from System.ComponentModel.Component
Equals	Inherited from System.Object (Overloaded)
GetDeleteCommand	Gets the automatically generated OracleCommand object that has the SQL statement (CommandText) perform deletions on the database (Overloaded)
GetHashCode	Inherited from System.Object
GetInsertCommand	Gets the automatically generated OracleCommand object that has the SQL statement (CommandText) perform insertions on the database (Overloaded)
GetLifetimeService	Inherited from System.MarshalByRefObject
GetType	Inherited from System.Object
GetUpdateCommand	Gets the automatically generated OracleCommand object that has the SQL statement (CommandText) perform updates on the database (Overloaded)
InitializeLifetimeService	Inherited from System.MarshalByRefObject
QuotedIdentifier	Returns the correct quoted form of the provided unquoted identifier, with any embedded quotes in the identifier properly escaped
RefreshSchema	Refreshes the database schema information used to generate INSERT, UPDATE, or DELETE statements
UnquotedIdentifier	Returns the correct unquoted form of the provided quoted identifier, removing any escape notation for quotes embedded in the identifier
ToString	Inherited from System.Object

OracleCommandBuilder Events

The OracleCommandBuilder event is listed in [Table 6-21](#).

Table 6-21 OracleCommandBuilder Events

Event Name	Description
Disposed	Inherited from System.ComponentModel.Component

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)

OracleCommandBuilder Constructors

OracleCommandBuilder constructors create new instances of the OracleCommandBuilder class.

Overload List:

- [OracleCommandBuilder\(\)](#)

This constructor creates an instance of the OracleCommandBuilder class.

- [OracleCommandBuilder\(OracleDataAdapter\)](#)

This constructor creates an instance of the OracleCommandBuilder class and sets the DataAdapter property to the provided OracleDataAdapter object.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)

OracleCommandBuilder()

This constructor creates an instance of the OracleCommandBuilder class.

Declaration

```
// C#  
public OracleCommandBuilder();
```

Remarks

Default constructor.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)

OracleCommandBuilder(OracleDataAdapter)

This constructor creates an instance of the OracleCommandBuilder class and sets the DataAdapter property to the provided OracleDataAdapter object.

Declaration

```
// C#  
public OracleCommandBuilder(OracleDataAdapter da);
```

Parameters

- da*
The `OracleDataAdapter` object provided.

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)

OracleCommandBuilder Static Methods

OracleCommandBuilder static methods are listed in [Table 6-22](#).

Table 6-22 OracleCommandBuilder Static Methods

Method	Description
DeriveParameters	Queries for the parameters of a stored procedure or function, represented by a specified <code>OracleCommand</code> , and populates the <code>OracleParameterCollection</code> of the command with the return values
<code>Equals</code>	Inherited from <code>System.Object</code> (Overloaded)

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)

DeriveParameters

This method queries for the parameters of a stored procedure or function, represented by a specified `OracleCommand`, and populates the `OracleParameterCollection` of the command with the return values.

Declaration

```
// C#  
public static void DeriveParameters(OracleCommand command);
```

Parameters

- command*

The command that represents the stored procedure or function for which parameters are to be derived.

Exceptions

`InvalidOperationException` - The `CommandText` is not a valid stored procedure or function name, the `CommandType` is not `CommandType.StoredProcedure`, or the `Connection.State` is not `ConnectionState.Open`.

Remarks

When `DeriveParameters` is used to populate the `Parameter` collection of an `OracleCommand` Object that represents a stored function, the return value of the function is bound as the first parameter (at position 0 of the `OracleParameterCollection`).

`DeriveParameters` can only be used for stored procedures or functions, not for anonymous PL/SQL blocks.

`DeriveParameters` incurs a database round-trip to retrieve parameter metadata prior to executing the stored procedure/function. It should only be used during design time. To avoid unnecessary database round-trips in a production environment, the `DeriveParameters` method itself should be replaced with the explicit parameter settings that were returned by the `DeriveParameters` method at design time.

`DeriveParameters` can only preserve the case of the stored procedure or function name if it is encapsulated by double-quotes. For example, if the stored procedure in the database is named `GetEmployees` with mixed-case, the `CommandText` property on the `OracleCommand` object must be set appropriately as in the following example:

```
cmd.CommandText = "\"GetEmployees\"";
```

Stored procedures and functions in a package must be provided in the following format:

```
<package name>.<procedure or function name>
```

For example, to obtain parameters for a stored procedure named `GetEmployees` (mixed-case) in a package named `EmpProcedures` (mixed-case), the name provided to the `OracleCommand` is:

```
"\"EmpProcedures\".\"GetEmployees\""
```

`DeriveParameters` cannot be used for object type methods.

The derived parameters contain all the metadata information that is needed for the stored procedure to execute properly. The application must provide the value of the parameters before execution, if required. The application may also modify the metadata information of the parameters before execution. For example, the `Size` property of the `OracleParameter` may be modified for PL/SQL character and string types to optimize the execution of the stored procedure.

The output values of derived parameters return as .NET Types by default. To obtain output parameters as provider types, the `OracleDbType` property of the parameter must be set explicitly by the application to override this default behavior. One quick way to do this is to set the `OracleDbType` to itself for all output parameters that should be returned as provider types.

The `BindByName` property of the supplied `OracleCommand` is left as is, but the application can change its value.

If the specified stored procedure or function is overloaded, the first overload is used to populate the parameters collection.

```
// Database Setup
/*
connect scott/tiger@oracle
CREATE OR REPLACE PROCEDURE MyOracleStoredProc (arg_in IN VARCHAR2,
  arg_out OUT VARCHAR2) IS
BEGIN
  arg_out := arg_in;
END;
/
*/

// C#
using System;
using System.Data;
using Oracle.DataAccess.Client;

class DeriveParametersSample
{
  static void Main()
  {
    // Create the PL/SQL Stored Procedure MyOracleStoredProc as indicated in
    // the preceding Database Setup

    string constr = "User Id=scott;Password=tiger;Data Source=oracle";
    OracleConnection con = new OracleConnection(constr);
    con.Open();

    // Create an OracleCommand
    OracleCommand cmd = new OracleCommand("MyOracleStoredProc", con);
    cmd.CommandType = CommandType.StoredProcedure;

    // Derive Parameters
    OracleCommandBuilder.DeriveParameters(cmd);
    Console.WriteLine("Parameters Derived");

    // Prints "Number of Parameters for MyOracleStoredProc = 2"
    Console.WriteLine("Number of Parameters for MyOracleStoredProc = {0}",
      cmd.Parameters.Count);

    // The PL/SQL stored procedure MyOracleStoredProc has one IN and
    // one OUT parameter. Set the Value for the IN parameter.
    cmd.Parameters[0].Value = "MyText";

    // The application may modify the other OracleParameter properties also
    // This sample uses the default Size for the IN parameter and modifies
    // the Size for the OUT parameter

    // The default size for OUT VARCHAR2 is 4000
    // Prints "cmd.Parameters[1].Size = 4000"
    Console.WriteLine("cmd.Parameters[1].Size = " + cmd.Parameters[1].Size);

    // Set the Size for the OUT parameter
    cmd.Parameters[1].Size = 6;

    // Execute the command
    cmd.ExecuteNonQuery();

    // Prints "cmd.Parameters[1].Value = MyText"
    Console.WriteLine("cmd.Parameters[1].Value = " + cmd.Parameters[1].Value);

    con.Close();
    con.Dispose();
  }
}
```

```
}
}
```

Example

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)
- [OracleCommand Class](#)
- [OracleParameter Class](#)
- [OracleParameterCollection Class](#)
- <http://msdn.microsoft.com/library> for detailed information about this Microsoft .NET Framework feature

OracleCommandBuilder Properties

OracleCommandBuilder properties are listed in [Table 6-23](#).

Table 6-23 OracleCommandBuilder Properties

Property	Description
Container	Inherited from System.ComponentModel.Component
CaseSensitive	Indicates whether or not double quotes are used around Oracle object names when generating SQL statements
CatalogLocation	<i>Not Supported</i>
CatalogSeparator	<i>Not Supported</i>
ConflictOption	<i>Not Supported</i>
DataAdapter	Indicates the OracleDataAdapter for which the SQL statements are generated
QuotePrefix	Specifies the beginning character or characters used to specify database objects whose names contain special characters such as spaces or reserved words
QuoteSuffix	Specifies the ending character or characters used to specify database objects whose names contain special characters such as spaces or reserved words
SchemaSeparator	Specifies the character to be used for the separator between the schema identifier and other identifiers
Site	Inherited from System.ComponentModel.Component

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)

CaseSensitive

This property indicates whether or not double quotes are used around Oracle object names (for example, tables or columns) when generating SQL statements.

Declaration

```
// C#  
bool CaseSensitive {get; set;}
```

Property Value

A bool that indicates whether or not double quotes are used.

Remarks

Default = false

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)

CatalogLocation

This property is not supported.

Declaration

```
// C#  
public override CatalogLocation CatalogLocation {get; set;}
```

Exceptions

NotSupportedException - This property is not supported.

Remarks

This property is not supported.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)

CatalogSeparator

This property is not supported.

Declaration

```
// C#  
public override string CatalogSeparator {get; set;}
```

Exceptions

NotSupportedException - This property is not supported.

Remarks

This property is not supported.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)

ConflictOption

This property is not supported.

Declaration

```
// C#  
public override string ConflictOption {get; set;}
```

Exceptions

NotSupportedException - This property is not supported.

Remarks

This property is not supported.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)

DataAdapter

This property indicates the `OracleDataAdapter` object for which the SQL statements are generated.

Declaration

```
// C#  
OracleDataAdapter DataAdapter{get; set;}
```

Property Value

An `OracleDataAdapter` object.

Remarks

Default = `null`

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)

QuotePrefix

This property specifies the beginning character or characters used to specify database objects whose names contain special characters such as spaces or reserved words.

Declaration

```
// C#  
public override string QuotePrefix {get; set;}
```

Property Value

The beginning character or characters to use. The default value is `"\"`.

Remarks

This property is independent of any `OracleConnection` or `OracleCommand` objects.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)

QuoteSuffix

This property specifies the ending character or characters used to specify database objects whose names contain special characters such as spaces or reserved words.

Declaration

```
// C#  
public override string QuoteSuffix {get; set;}
```

Property Value

The ending character or characters to use. The default value is "\"".

Remarks

This property is independent of any `OracleConnection` or `OracleCommand` objects.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)

SchemaSeparator

This property specifies the character to be used for the separator between the schema identifier and other identifiers.

Declaration

```
// C#  
public override string SchemaSeparator {get; set; }
```

Property Value

The character to be used as the schema separator.

Exceptions

`NotSupportedException` - The input value is not a dot (.).

Remarks

The default schema separator is a dot (.). The only acceptable value for this property is a dot (.).

This property is independent of any `OracleConnection` or `OracleCommand` objects.

Example

```
// C#

using System;
using System.Data;
using System.Data.Common;
using Oracle.DataAccess.Client;

class SchemaSeperatorSample
{
    static void Main(string[] args)
    {
        try
        {
            OracleCommandBuilder cmdBuilder = new OracleCommandBuilder();

            //schemaSeparator is dot(.)
            Console.WriteLine("schemaSeparator is {0}",
                             cmdBuilder.SchemaSeparator);

            //set the schemaseparator, only '.' is allowed.
            cmdBuilder.SchemaSeparator = ".";

            // the only acceptable value for this property is a dot (.)
            // Hence the following line will throw NotSupportedException
            cmdBuilder.SchemaSeparator = "!";
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
            Console.WriteLine(ex.StackTrace);
        }
    }
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)

OracleCommandBuilder Public Methods

`OracleCommandBuilder` public methods are listed in [Table 6-24](#).

Table 6-24 OracleCommandBuilder Public Methods

Public Method	Description
CreateObjRef	Inherited from System.MarshalByRefObject
Dispose	Inherited from System.ComponentModel.Component
Equals	Inherited from System.Object (Overloaded)
GetDeleteCommand	Gets the automatically generated OracleCommand object that has the SQL statement (CommandText) perform deletions on the database (Overloaded)
GetHashCode	Inherited from System.Object
GetInsertCommand	Gets the automatically generated OracleCommand object that has the SQL statement (CommandText) perform insertions on the database (Overloaded)
GetLifetimeService	Inherited from System.MarshalByRefObject
GetType	Inherited from System.Object
GetUpdateCommand	Gets the automatically generated OracleCommand object that has the SQL statement (CommandText) perform updates on the database (Overloaded)
InitializeLifetimeService	Inherited from System.MarshalByRefObject
QuoteIdentifier	Returns the correct quoted form of the provided unquoted identifier, with any embedded quotes in the identifier properly escaped
RefreshSchema	Refreshes the database schema information used to generate INSERT, UPDATE, or DELETE statements
UnquoteIdentifier	Returns the correct unquoted form of the provided quoted identifier, removing any escape notation for quotes embedded in the identifier
ToString	Inherited from System.Object

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)

GetDeleteCommand

Gets the automatically generated OracleCommand object that has the SQL statement (CommandText) perform deletions on the database

Overload List

- [GetDeleteCommand\(\)](#)

This method gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform deletions on the database when an application calls `Update()` on the `OracleDataAdapter`.

- [GetDeleteCommand\(bool\)](#)

This method gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform deletions on the database when an application calls `Update()` on the `OracleDataAdapter`.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)

GetDeleteCommand()

This method gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform deletions on the database when an application calls `Update()` on the `OracleDataAdapter`.

Declaration

```
// C#  
public OracleCommand GetDeleteCommand();
```

Return Value

An `OracleCommand`.

Exceptions

`ObjectDisposedException` - The `OracleCommandBuilder` object is already disposed.

`InvalidOperationException` - Either the `SelectCommand` or the `DataAdapter` property is null, or the primary key cannot be retrieved from the `SelectCommand` property of the `OracleDataAdapter`.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)

GetDeleteCommand(bool)

This method gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform deletions on the database when an application calls `Update()` on the `OracleDataAdapter`.

Declaration

```
// C#  
public OracleCommand GetDeleteCommand(bool useColumnsForParameterNames);
```

Parameters

- `useColumnsForParameterNames`

If true, the method generates parameter names matching column names if possible. If false, the method binds parameters by position.

Return Value

An `OracleCommand`.

Exceptions

`ObjectDisposedException` - The `OracleCommandBuilder` object is already disposed.

`InvalidOperationException` - Either the `SelectCommand` or the `DataAdapter` property is null, or the primary key cannot be retrieved from the `SelectCommand` property of the `OracleDataAdapter`.

Remarks

If the `bool` is true, the method generates parameter names matching column names if possible. If false, the method binds parameters by position.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)

GetInsertCommand

Gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform insertions on the database

Overload List

- [GetInsertCommand\(\)](#)

This method gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform insertions on the database when an application calls `Update()` on the `OracleDataAdapter`.

- [GetInsertCommand\(bool\)](#)

This method gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform insertions on the database when an application calls `Update()` on the `OracleDataAdapter`.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)

GetInsertCommand()

This method gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform insertions on the database when an application calls `Update()` on the `OracleDataAdapter`.

Declaration

```
// C#  
public OracleCommand GetInsertCommand();
```

Return Value

An `OracleCommand`.

Exceptions

`ObjectDisposedException` - The `OracleCommandBuilder` object is already disposed.

`InvalidOperationException` - Either the `SelectCommand` or the `DataAdapter` property is null, or the primary key cannot be retrieved from the `SelectCommand` property of the `OracleDataAdapter`.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)

GetInsertCommand(bool)

This method gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform insertions on the database when an application calls `Update()` on the `OracleDataAdapter`.

Declaration

```
// C#  
public OracleCommand GetInsertCommand(bool useColumnsForParameterNames);
```

Parameters

- `useColumnsForParameterNames`

If true, the method generates parameter names matching column names if possible. If false, the method binds parameters by position.

Return Value

An `OracleCommand`.

Exceptions

`ObjectDisposedException` - The `OracleCommandBuilder` object is already disposed.

`InvalidOperationException` - Either the `SelectCommand` or the `DataAdapter` property is null, or the primary key cannot be retrieved from the `SelectCommand` property of the `OracleDataAdapter`.

Remarks

If the `bool` is true, the method generates parameter names matching column names if possible. If false, the method binds parameters by position.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)

GetUpdateCommand

Gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform updates on the database

Overload List

- [GetUpdateCommand\(\)](#)

This method gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform updates on the database when an application calls `Update()` on the `OracleDataAdapter`.

- [GetUpdateCommand\(bool\)](#)

This method gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform updates on the database when an application calls `Update()` on the `OracleDataAdapter`.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)

GetUpdateCommand()

This method gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform updates on the database when an application calls `Update()` on the `OracleDataAdapter`.

Declaration

```
// C#  
public OracleCommand GetUpdateCommand();
```

Return Value

An `OracleCommand`.

Exceptions

`ObjectDisposedException` - The `OracleCommandBuilder` object is already disposed.

`InvalidOperationException` - Either the `SelectCommand` or the `DataAdapter` property is null, or the primary key cannot be retrieved from the `SelectCommand` property of the `OracleDataAdapter`.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)

GetUpdateCommand(bool)

This method gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform updates on the database when an application calls `Update()` on the `OracleDataAdapter`.

Declaration

```
// C#  
public OracleCommand GetUpdateCommand(bool useColumnsForParameterNames);
```

Parameters

- `useColumnsForParameterNames`

If true, the method generates parameter names matching column names if possible. If false, the method binds parameters by position.

Return Value

An `OracleCommand`.

Exceptions

`ObjectDisposedException` - The `OracleCommandBuilder` object is already disposed.

InvalidOperationException - Either the SelectCommand or the DataAdapter property is null, or the primary key cannot be retrieved from the SelectCommand property of the OracleDataAdapter.

Remarks

If the bool is true, the method generates parameter names matching column names if possible. If false, the method binds parameters by position.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)

QuoteIdentifier

This method returns the correct quoted form of the provided unquoted identifier, with any embedded quotes in the identifier properly escaped.

Declaration

```
// C#  
public override string QuoteIdentifier(string unquotedIdentifier);
```

Parameters

- *UnquotedIdentifier*
An unquoted identifier string.

Return Value

The quoted version of the identifier. Embedded quotes within the identifier are properly escaped.

Exceptions

ArgumentNullException - The input parameter is null.

Remarks

This method is independent of any OracleConnection or OracleCommand objects.

Example

```
// C#  
  
using System;  
using System.Data;  
using System.Data.Common;  
using Oracle.DataAccess.Client;  
  
class QuoteIdentifierSample  
{  
    static void Main(string[] args)
```

```
{
    OracleCommandBuilder builder = new OracleCommandBuilder();
    string quoteIdentifier = builder.QuoteIdentifier("US\"ER");

    //quoteIdentifier for "US\"ER" is (\\"US\\"ER\\")
    Console.WriteLine("quoteIdentifier is {0}" , quoteIdentifier);
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)

RefreshSchema

This method refreshes the database schema information used to generate INSERT, UPDATE, or DELETE statements.

Declaration

```
// C#
public override void RefreshSchema();
```

Remarks

An application should call RefreshSchema whenever the SelectCommand value of the OracleDataAdapter object changes.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)

UnquoteIdentifier

This method returns the correct unquoted form of the provided quoted identifier, removing any escape notation for quotes embedded in the identifier.

Declaration

```
// C#
public override string UnquoteIdentifier(string quotedIdentifier);
```

Parameters

- *quotedIdentifier*
The quoted string identifier.

Return Value

The unquoted identifier, with escape notation for any embedded quotes removed.

Exceptions

ArgumentNullException - The input parameter is null.

ArgumentException - The input parameter is empty.

Remarks

This method is independent of any OracleConnection or OracleCommand objects.

Example

```
// C#

using System;
using System.Data;
using System.Data.Common;
using Oracle.DataAccess.Client;

class UnQuoteIdentifierSample
{
    static void Main(string[] args)
    {
        //create an OracleCommandBuilder object.
        OracleCommandBuilder builder = new OracleCommandBuilder();

        string identifier = "US\"ER";
        Console.WriteLine("Identifier is {0}", identifier);

        // quote the identifier
        string quoteIdentifier = builder.QuoteIdentifier(identifier);

        //quoteIdentifier of "US\"ER" is (\\"US\\"ER\\")
        Console.WriteLine("QuotedIdentifier is {0}" , quoteIdentifier);
        string unquoteIdentifier = builder.UnquoteIdentifier(quoteIdentifier);

        //And its unquoteIdentifier is US\"ER
        Console.WriteLine("UnquotedIdentifier is {0}" , unquoteIdentifier);
    }
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)

OracleCommandBuilder Events

The OracleCommandBuilder event is listed in [Table 6-25](#).

Table 6-25 OracleCommandBuilder Event

Event Name	Description
Disposed	Inherited from System.ComponentModel.Component

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommandBuilder Class](#)
- [OracleCommandBuilder Members](#)

OracleConfiguration Class

An OracleConfiguration is a static class for setting ODP.NET configuration data using a single programming interface.

Class Inheritance

System.Object

Oracle.ManagedDataAccess.Client.OracleConfiguration

Declaration

```
// C#  
public sealed class OracleConfiguration
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Note

Unmanaged ODP.NET implements the OracleConfiguration debug tracing properties only.

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#

using System;
using Oracle.ManagedDataAccess.Client;

namespace ODP_Core_Config_API
{
    class odp_core_config
    {
        static void Main(string[] args)
        {
            // This sample demonstrates how to use ODP.NET Core Configuration API

            // Add connect descriptors and net service names entries.
            OracleConfiguration.OracleDataSources.Add("orclpdb",
                "(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<hostname or IP>)(PORT=1521))
                (CONNECT_DATA=(SERVICE_NAME=<service name>)(SERVER=dedicated))))");
            OracleConfiguration.OracleDataSources.Add("orcl",
                "(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<hostname or IP>)(PORT=1521))
                (CONNECT_DATA=(SERVICE_NAME=<service name>)(SERVER=dedicated))))");

            // Set default statement cache size to be used by all connections.
            OracleConfiguration.StatementCacheSize = 25;

            // Disable self tuning by default.
            OracleConfiguration.SelfTuning = false;

            // Bind all parameters by name.
            OracleConfiguration.BindByName = true;

            // Set default timeout to 60 seconds.
            OracleConfiguration.CommandTimeout = 60;

            // Set default fetch size as 1 MB.
            OracleConfiguration.FetchSize = 1024 * 1024;

            // Set tracing options
            OracleConfiguration.TraceOption = 1;
            OracleConfiguration.TraceFileLocation = @"D:\traces";
            // Uncomment below to generate trace files
            //OracleConfiguration.TraceLevel = 7;

            // Set network properties
            OracleConfiguration.SendBufferSize = 8192;
            OracleConfiguration.ReceiveBufferSize = 8192;
            OracleConfiguration.DisableO0B = true;

            OracleConnection orclCon = null;

            try
            {
                // Open a connection
                orclCon = new OracleConnection("user id=hr; password=<password>; data
```

```

source=orclpdb");
    orclCon.Open();

    // Execute simple select statement that returns first 10 names from
EMPLOYEES table
    OracleCommand orclCmd = orclCon.CreateCommand();
    orclCmd.CommandText = "select first_name from employees where rownum <=
10 ";

    OracleDataReader rdr = orclCmd.ExecuteReader();

    while (rdr.Read())
        Console.WriteLine("Employee Name: " + rdr.GetString(0));

    Console.ReadLine();

    rdr.Dispose();
    orclCmd.Dispose();
}
finally
{
    // Close the connection
    if (null != orclCon)
        orclCon.Close();
}
}
}
}

```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Members](#)
- [OracleConfiguration Methods](#)
- [OracleConfiguration Connection Properties](#)
- [OracleConfiguration Secure Connection Properties](#)
- [OracleConfiguration Command Properties](#)
- [OracleConfiguration High Availability Properties](#)
- [OracleConfiguration Performance Properties](#)
- [OracleConfiguration Diagnostics and Tracing Properties](#)

OracleConfiguration Members

OracleConfiguration members are listed in the following tables.

OracleConfiguration Methods

OracleConfiguration methods are listed in [Table 6-26](#).

Table 6-26 OracleConfiguration Methods

Method	Description
DisableConnectionStringAllowedProperties	Disables connection configuration restriction <i>Not available in ODP.NET, Unmanaged Driver</i>
EnableConnectionStringAllowedProperties	Accepts an allowed connection configuration parameter list (Overloaded) <i>Not available in ODP.NET, Unmanaged Driver</i>

OracleConfiguration Connection Properties

OracleConfiguration Connection properties are listed in [Table 6-27](#).

Table 6-27 OracleConfiguration Connection Properties

Property	Description
ConfigurationProviders	Enables one or more centralized configuration providers <i>Not available in ODP.NET, Unmanaged Driver</i>
DatabaseCharset	Sets the CHAR database character set at the process level and is used to optimize connection establishment <i>Not available in ODP.NET, Unmanaged Driver</i>
DatabaseNCharset	Sets the NCHAR database character set at the process level and is used to optimize connection establishment <i>Not available in ODP.NET, Unmanaged Driver</i>
DatabaseEditionName	Specifies the Oracle edition name for the connection object
DefaultSDUSize	Specifies the session data unit size (SDU) for Oracle networking communication between client and server
DisableOOB	Specifies whether to enable Oracle Net Services to send or receive out-of-band break messages using urgent data provided by the underlying protocol
DrpcConnectionClass	Specifies a logical name that identifies the DRCP connection pool that the ODP.NET connection will use
HostnameDefaultServiceIsHost	Specifies whether to default the service name to the hostname in Oracle Easy Connect Naming so that the service name does not need to be specified
NamesDirectoryPath	Specifies the naming method order used for client name resolution lookups
OciCompartment	Gets or sets the Oracle Cloud Identifier (OCID) of the target database's compartment at the process level
OciConfigurationFile	Gets or sets the configuration file location at the process level where the OCI token authentication values can be found. <i>Not available in ODP.NET, Unmanaged Driver</i>
OciDatabase	Gets or sets the target database's Oracle Cloud Identifier (OCID) at the process level
OciIamUrl	Gets or sets the URL end point at the process level for getting the database token

Table 6-27 (Cont.) OracleConfiguration Connection Properties

Property	Description
OciProfile	Gets or sets the OCI profile at the process level to use from the token authentication configuration file. <i>Not available in ODP.NET, Unmanaged Driver</i>
OciTenacy	Gets or sets the Oracle Cloud Identifier (OCID) of the IAM user's cloud tenancy at the process level for retrieving the database token
OracleDataSources	Returns a collection of TNS entries added through OracleConfiguration
SuppressGetDecimalInvalidCastException	Specifies whether to suppress the InvalidCastException and return a rounded-off 28 or 29 precision Oracle NUMBER value that can be represented as a .NET Decimal <i>Not available in ODP.NET, Unmanaged Driver</i>
TAFUsePooledConnections	Indicates whether to use pre-existing connections from the pool for failover <i>Not available in ODP.NET, Unmanaged Driver</i>
TcpConnectTimeout	Specifies the time, in seconds, for a client to establish a TCP connection (PROTOCOL=tcp in the TNS connect address) to the database server before it can time out
TimeZoneFileLocation	Specifies the time zone file directory
TnsAdmin	Specifies the tnsnames.ora and/or sqlnet.ora directory location
UseSNI	Specifies whether to use Server Name Indication (SNI) <i>Not Available in ODP.NET, Unmanaged Driver</i>

OracleConfiguration Secure Connection Properties

OracleConfiguration Secure Connection Properties are listed in [Table 6-28](#).

Table 6-28 OracleConfiguration Secure Connection Properties

Property	Description
AllowUncPaths	Allows or disallows ODP.NET access to configuration files with Universal Naming Convention location paths <i>Not available in ODP.NET, Unmanaged Driver</i>
PasswordAuthentication	Sets the process level PASSWORD_AUTH parameter value
SEPSWalletLocation	Specifies the Secure External Password Store wallet location, typically used for PKCS #12 wallet files
SEPSWalletPassword	Stores the password to decrypt the PKCS #12 wallet located in the Secure External Password Store
SqlNetAllowWeakCrypto	Allows or disallows weaker encryption and checksum algorithms for database connections
SqlNetAllowedLogonVersionClient	Specifies the minimum authentication protocol that is to be used when authenticating against the Oracle database for a given application

Table 6-28 (Cont.) OracleConfiguration Secure Connection Properties

Property	Description
SqlNetAuthenticationServices	Enables one or more authentication services, such as TCP/IP with SSL
SqlNetCloudUser	Specifies an user name for web server HTTP basic authentication
SqlNetCryptoChecksumClient	Specifies the checksum client behavior
SqlNetCryptoChecksumTypesClient	Specifies the crypto-checksum algorithms the client can use
SqlNetEncryptionClient	Specifies the encryption client behavior
SqlNetEncryptionTypesClient	Specifies encryption algorithms that the client can use
SqlNetIgnoreANOEncryptionForTCPS	Specifies whether to disable double encryption and double checksumming under certain conditions <i>Not available in ODP.NET, Unmanaged Driver</i>
SqlNetWalletOverride	Specifies whether the client overrides the strong authentication credential with the stored wallet password credential for the database connection
SqlNetURI	Specifies the WebSocket universal resource identifier
SSLCertificateThumbprint	Specifies the TLS/SSL certificate thumbprint in the certificate store for the ODP.NET connection to use
SSLServerDNMatch	Enforces the database server distinguished name (DN) matches its service name
SSLVersion	Enforces the connection to use a specific SSL/TLS version
TLSExcludeVersion	Indicates which SSL/TLS versions connections should not be enabled for <i>Not available in ODP.NET, Unmanaged Driver</i>
TLSVersion	Enforces the connection to use a specific SSL/TLS version <i>Not available in ODP.NET, Unmanaged Driver</i>
TokenAuthentication	Sets the value for TOKEN_AUTH parameter at the process level
TokenLocation	Gets or sets the token location directory at the process level where the access token , or database token and private key files, are present for ODP.NET token authentication.
UseMITKerberos	Specifies whether to use MIT Kerberos or Kerberos.NET <i>Only available in ODP.NET, Managed Driver</i>
WalletLocation	Specifies the location of wallets
WalletPassword	Stores the password to decrypt the PKCS #12 wallet <i>Not available in ODP.NET, Unmanaged Driver</i>

OracleConfiguration Command Properties

OracleConfiguration Command properties are listed in [Table 6-29](#).

Table 6-29 OracleConfiguration Command Properties

Property	Description
AddOracleTypesDeserialization	Adds ODP.NET-specific data types to the “allow” list to permit deserialization into DataSet or DataTable
BindByName	Specifies whether the binding method used for the parameter collection is by name or by position
CommandTimeout	Specifies the number of seconds the command is allowed to execute before terminating the execution with an exception
GetDecimalRetainTrailingZeros	Specifies whether to retain trailing zeros from an Oracle NUMBER in a .NET Decimal
MapVectorColumnAsClob	Specifies whether to retrieve vector types as native vectors or CLOB/VARCHAR2 in .NET <i>Not available in ODP.NET, Unmanaged Driver</i>
ProviderTypeNLSSensitiveToString	Specifies whether to honor the data type's NLS format when the ToString() method is called <i>Not available in ODP.NET, Unmanaged Driver</i>
UseClientInitiatedCQN	Specifies whether to use Client Initiated Continuous Query Notification (CICQN) or traditional Continuous Query Notification (CQN) that does not rely on a persistent connection

OracleConfiguration Directories Properties

OracleConfiguration Directories properties are listed in [Table 6-30](#).

Table 6-30 OracleConfiguration Directories Properties

Property	Description
DefaultAdminContext	Specifies the default directory entry that contains a LDAP Oracle Context from which connect identifiers can be created, modified, or looked up
DirectoryServers	Lists the host names and port numbers of the primary and alternate LDAP directory servers
DirectoryServerType	Specifies the directory server type used, such as Oracle Internet Directory or Microsoft Active Directory
LdapAdmin	Specifies the ldap.ora directory location
NamesLdapAuthenticateBind	Specifies whether the LDAP naming adapter should attempt to authenticate using a specified wallet when it connects to the LDAP directory to resolve the name in the connect string
NamesLdapAuthenticateBindMethod	Specifies the authentication method the client LDAP naming adapter should use while connecting to the LDAP directory to resolve connect string names
NamesLdapConnTimeout	specifies the non-blocking connection timeout to the LDAP server

OracleConfiguration High Availability Properties

OracleConfiguration High Availability properties are listed in [Table 6-31](#).

Table 6-31 OracleConfiguration High Availability Properties

Property	Description
ChunkMigrationConnectionTimeout	This setting pauses the connection request until the migration completes or fails to migrate in the specified time, then connect to the correct shard with the data requested. <i>Not Available in ODP.NET, Unmanaged Driver</i>
DbNotificationAddress	Specifies the address that the provider listens to for all notifications sent by the database for continuous query notification feature.
DbNotificationPort	Specifies the port number that the provider listens to for all notifications sent by the database for continuous query notification, Fast Connection Failover, and/or Runtime Connection Load Balancing features
DistributedTransactionRecoveryAddress	Specifies the machine name or IP address the OraMTS recovery service runs on to resolve database in-doubt transactions
DistributedTransactionRecoveryPort	Specifies the port the OraMTS recovery service listens on to resolve database in-doubt transactions
DistributedTransactionTimeToLive	Specifies the time the transaction can remain inactive after it has been detached or delisted from the database
HAEvents	Enables the application to receive Fast Connection Failover events for maintaining application high availability
MaxCallsPerRequestBoundary	Specifies the maximum request size, or ODP.NET calls, beyond which Application Continuity replay will be disabled
OnsConfigFile	Specifies the configuration file to define Oracle Notification Service (ONS) behavior
OnsMode	Specifies the ONS daemon mode as either local or remote
OnsProtocol	Defines whether the ONS connection uses TCP/IP or TCP/IP with SSL/TLS
OnsWalletLocation	Specifies the directory location of the ONS wallets
OracleOnsServers	Returns a collection of logical ONS servers added through OracleConfiguration
ServiceRelocationConnectionTimeout	Specifies the time to wait before retrying connecting to a service that becomes unavailable.

OracleConfiguration Performance Properties

OracleConfiguration Performance properties are listed in [Table 6-32](#).

Table 6-32 OracleConfiguration Performance Properties

Property	Description
FetchSize	Specifies the total memory size, in bytes, that the provider allocates to cache data fetched in one database round-trip
LoadBalancing	Enables the application to receive runtime connection load balancing information
MaxStatementCacheSize	Specifies the maximum number of statements that can be cached when self-tuning is enabled

Table 6-32 (Cont.) OracleConfiguration Performance Properties

Property	Description
Pipelining	Enables asynchronous execution on the database server side
PerformanceCounters	Specifies the ODP.NET connection performance counters to publish so that they can be monitored
ReceiveBufferSize	Specifies the buffer space limit for receive operations of sessions
RowsToFetchPerRoundTrip	Specifies the total number of rows to retrieve per database round trip <i>Not Available in ODP.NET, Unmanaged Driver</i>
SelfTuning	Specifies whether self-tuning is enabled for an ODP.NET application
SendBufferSize	Specifies the buffer space limit for send operations of sessions
StatementCacheSize	Specifies the number of cursors or statements to be cached for each database connection
TcpNoDelay	Preempts delays in buffer flushing within the TCP/IP protocol stack

OracleConfiguration Diagnostics and Tracing Properties

OracleConfiguration Diagnostics and Tracing properties are listed in [Table 6-33](#).

Table 6-33 OracleConfiguration Diagnostics and Tracing Properties

Property	Description
ConnectionIdPrefix	Identifies a particular application's connections by appending its string to the connection identifier <i>Not Available in ODP.NET, Unmanaged Driver</i>
OpenTelemetryTracing	Specifies whether to dynamically enable or disable ODP.NET OpenTelemetry tracing instrumentation or not. <i>Not Available in ODP.NET, Unmanaged Driver</i>
PoolNames	Specifies a string to identify the monitored connection pool more easily
Program	Specifies the application program name <i>Not available in ODP.NET, Unmanaged Driver</i>
SuppressErrorURL	Shows or suppresses the Oracle Database Error Messages URL for the specific error with more information on its cause and resolution
TraceFileLocation	Specifies the destination directory to output provider traces
TraceFileMaxSize	Specifies the maximum file size of each trace file
TraceLevel	Specifies the generated trace level to trace ODP.NET calls and diagnose provider issues
TraceOption	Specifies whether to generate a single trace file or multiple trace files for multithreaded applications

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)

OracleConfiguration Methods

OracleConfiguration methods are listed in [Table 6-34](#).

Table 6-34 OracleConfiguration Methods

Method	Description
DisableConnectionStringAllowedProperties	Disables connection configuration restriction <i>Not available in ODP.NET, Unmanaged Driver</i>
EnableConnectionStringAllowedProperties	Accepts an allowed connection configuration parameter list (Overloaded) <i>Not available in ODP.NET, Unmanaged Driver</i>

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

DisableConnectionStringAllowedProperties

This method disables connection configuration restriction.

Declaration

```
// C#  
public static void DisableConnectionStringAllowedProperties()
```

Exceptions

None.

Remarks

When invoked, any prior list of allowed parameters is discarded. Connections are no longer subject to configuration restrictions.

Connection configuration restriction can be enabled again by calling an `EnableConnectionStringAllowedProperties` method. Calling this method does not impact existing connection pools until `ClearPool()` or `ClearAllPools()` is called.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)
- [Connection Configuration Restriction](#)

EnableConnectionStringAllowedProperties

Overload List:

- [EnableConnectionStringAllowedProperties\(string, bool\)](#)
This method accepts an allowed connection configuration parameter list in a JSON string format.
- [EnableConnectionStringAllowedProperties\(FileInfo, bool\)](#)
This method accepts a file path to an allowed parameter list in a JSON file.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

EnableConnectionStringAllowedProperties(string, bool)

This method accepts an allowed connection configuration parameter list in a JSON string format.

Declaration

```
// C#  
public static void EnableConnectionStringAllowedProperties(string allowList, bool  
enforceOnAlias)
```

Parameters

- *allowList*
Allowed parameters list in JSON string format
- *enforceOnAlias*
Whether to enforce the allowed parameters list on TNS aliases

Exceptions

None.

Remarks

ODP.NET will check the connection string Data Source if any disallowed parameters are included. If so, the request will not be allowed to connect.

Applications can invoke this `EnableConnectionStringAllowedProperties` method anytime during runtime with a new allowed list. The new settings will apply to new non-pooled connections and new connection pools created henceforth.

For existing connection pools, the new settings are applied after `ClearPool()` or `ClearAllPools()` is invoked.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)
- [Connection Configuration Restriction](#) for JSON string format, examples, and details

EnableConnectionStringAllowedProperties(FileInfo, bool)

This method accepts a file path to an allowed parameter list in a JSON file.

Declaration

```
// C#
public static void EnableConnectionStringAllowedProperties(FileInfo jsonFile, bool
enforceOnAlias)
```

Parameters

- *jsonFile*
Specifies the JSON allowed parameters file path.
- *enforceOnAlias*
Whether to enforce the allowed parameters list on TNS aliases

Exceptions

`OracleException` – If Universal Naming Convention (UNC) file paths are not allowed and the file path is configured with a UNC file path.

`OracleException` – An `OracleException` (ORA-50124: Configuration file is required to be named as {0}) is thrown if the file path contains a file name other than `ODPConnectionStringAllowedProperties.json`.

`InvalidOperationException` – If this method is invoked with a different JSON file path than used earlier after at least one connection has been opened.

Remarks

The JSON file parameter of type FileInfo can be set to the directory that contains the ODPConnectionStringAllowedProperties.json file or to the full file path with that JSON file. If a file name is supplied, it must be ODPConnectionStringAllowedProperties.json.

ODP.NET will check the connection string Data Source if any disallowed parameters are included. If so, the request will not be allowed to connect.

If the application calls the EnableConnectionStringAllowedProperties method before the first connection is created, then ODP.NET tries to open and read the ODPConnectionStringAllowedProperties.json file. Failure to open or read the file throws an exception.

The application cannot invoke the EnableConnectionStringAllowedProperties method, which takes a different JSON file location after a valid connection has been opened. However, the application may invoke the method again with the same file path and a different enforceOnAlias value. The Boolean parameter will be applied as configured.

The application can update the JSON file contents during runtime. ODP.NET proactively tries to read the file once it is saved.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)
- [Connection Configuration Restriction](#) for JSON string format and examples

OracleConfiguration Connection Properties

OracleConfiguration Connection properties are listed in [Table 6-35](#).

Table 6-35 OracleConfiguration Connection Properties

Property	Description
ConfigurationProviders	Enables one or more centralized configuration providers <i>Not available in ODP.NET, Unmanaged Driver</i>
DatabaseCharset	Sets the CHAR database character set at the process level and is used to optimize connection establishment <i>Not available in ODP.NET, Unmanaged Driver</i>
DatabaseNCharset	Sets the NCHAR database character set at the process level and is used to optimize connection establishment <i>Not available in ODP.NET, Unmanaged Driver</i>
DatabaseEditionName	Specifies the Oracle edition name for the connection object
DisableOOB	Specifies whether to enable Oracle Net Services to send or receive out-of-band break messages using urgent data provided by the underlying protocol

Table 6-35 (Cont.) OracleConfiguration Connection Properties

Property	Description
DrpcConnectionClass	Specifies a logical name that identifies the DRCP connection pool that the ODP.NET connection will use
HostnameDefaultServiceIsHost	Specifies whether to default the service name to the hostname in Oracle Easy Connect Naming so that the service name does not need to be specified
NamesDirectoryPath	Specifies the naming method order used for client name resolution lookups
OciCompartment	Gets or sets the Oracle Cloud Identifier (OCID) of the target database's compartment at the process level
OciConfigurationFile	Gets or sets the configuration file location at the process level where the OCI token authentication values can be found. <i>Not available in ODP.NET, Unmanaged Driver</i>
OciDatabase	Gets or sets the target database's Oracle Cloud Identifier (OCID) at the process level
OciIamUrl	Gets or sets the URL end point at the process level for getting the database token
OciProfile	Gets or sets the OCI profile at the process level to use from the token authentication configuration file. <i>Not available in ODP.NET, Unmanaged Driver</i>
OciTenacy	Gets or sets the Oracle Cloud Identifier (OCID) of the IAM user's cloud tenancy at the process level for retrieving the database token
OracleDataSources	Returns a collection of TNS entries added through OracleConfiguration
SuppressGetDecimalInvalidCastException	Specifies whether to suppress the InvalidCastException and return a rounded-off 28 or 29 precision Oracle NUMBER value that can be represented as a .NET Decimal <i>Not available in ODP.NET, Unmanaged Driver</i>
TAFUsePooledConnections	Indicates whether to use pre-existing connections from the pool for failover <i>Not available in ODP.NET, Unmanaged Driver</i>
TcpConnectTimeout	Specifies the time, in seconds, for a client to establish a TCP connection (PROTOCOL=tcp in the TNS connect address) to the database server before it can time out
TimeZoneFileLocation	Specifies the time zone file directory
TnsAdmin	Specifies the tnsnames.ora and/or sqlnet.ora directory location
UseSNI	Specifies whether to use Server Name Indication (SNI) <i>Not Available in ODP.NET, Unmanaged Driver</i>

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

ConfigurationProviders

This property enables one or more centralized configuration providers.

Declaration

```
// C#  
public static string ConfigurationProviders { get; set; }
```

Property Type

System.String

Exception

- **ArgumentException:** Thrown when ConfigurationProviders contains an invalid configuration provider, or None is used with another provider, or the string is otherwise not properly formatted.
- **ArgumentNullExcpetion:** Thrown when ConfigurationProviders is set to null

Remarks

This property allows applications to enforce restrictions on which providers can be used.

Table 6-36 ConfigurationProviders Values

Value	Description
All	Enables all configuration providers. This is the default value.
None	Disables all configuration providers
Azure	Enables Azure App configuration provider
AzureVault	Enables Azure Vault Configuration provider
OCIObject	Enables Oracle Cloud Infrastructure (OCI) Configuration provider
OCIVault	Enables OCI Vault Configuration provider
File	Enable local file provider

The values are case-insensitive.

These values must be set prior to opening a connection with a centralized configuration provider. They cannot be modified after a connection is opened. The values are not verified until a URL is used upon the connection open.

Any combination of configuration providers can be enabled.

```
OracleConfiguration.ConfigurationProviders = "File, OCIObject";
```

Optionally, the list of providers can be enclosed in parentheses. For example, the following code sample enables Azure App configuration provider and OCI Configuration provider.

```
OracleConfiguration.ConfigurationProviders = "(Azure, OCIObject)";
```

Duplicate values are ignored, such as (Azure, OCIObject, Azure), and do not create an exception.

Using None in a list with other providers throws an exception, such as (All, None) or (None, OCIVault).

Setting the value to empty parentheses () will be the same as using None. However, using an empty string is invalid, and will result in an error when the value is verified.

Using All in a list with other providers enables all providers, such as (File, Azure, All).

For JSON-based providers, the providers used within the JSON config must also be enabled. For example, if the file provider uses an OCI Vault URL to store the password, both File and OCIVault must be enabled.

This setting can also be used in sqlnet.ora or .NET configuration file. The precedence order, from highest to lowest, is:

1. OracleConfiguration.ConfigurationParameters
2. .NET Config
3. sqlnet.ora

Alternatively, instead of using a string value, use the OracleConfigurationProvider enumeration and bitwise the enums to enable multiple providers. Only OracleConfiguration.ConfigurationProviders property allows this. Sqlnet.ora CONFIGURATION_PROVIDERS and .NET configuration file ConfigurationProviders do not support using this enumeration. They only support string values. Here is a code sample using ConfigurationProviders to enable the Azure App Configuration, OCI Object, and Azure Vault providers.

```
OracleConfiguration.ConfigurationProviders = OracleConfigurationProvider.Azure |  
OracleConfigurationProvider.OCIObject | OracleConfigurationProvider.AzureVault;
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

DatabaseCharset

This property sets the CHAR database character set at the process level and is used to optimize connection establishment.

Declaration

```
// C#  
public static OracleDatabaseCharset DatabaseCharset {set;}
```

Remarks

The default value is `OracleDatabaseCharset.AL32UTF8`.

When the application provides the actual CHAR database character set to connect with, the connection can be established in an optimized way with fewer roundtrips. If the property contains a non-matching character set value, connection establishment follows a traditional, less optimized process.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)
- [OracleDatabaseCharset Enumeration](#)

DatabaseNCharset

This property sets the NCHAR database character set at the process level and is used to optimize connection establishment.

Declaration

```
// C#  
public static OracleDatabaseNCharset DatabaseNCharset {set;}
```

Remarks

The default value is `OracleDatabaseNCharset.AL16UTF16`.

When the application provides the actual NCHAR database character set to connect with, the connection can be established in an optimized way with fewer roundtrips. If the property contains a non-matching character set value, connection establishment follows a traditional, less optimized process.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)
- [OracleDatabaseNCharset Enumeration](#)

DatabaseEditionName

This property specifies the Oracle edition name for the connection object. This property is used with the Oracle Edition-Based Redefinition feature.

Declaration

```
// C#  
public static string DatabaseEditionName { get; set; }
```

Property Type

System.String

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

DisableOOB

This property specifies whether to enable Oracle Net Services to send or receive out-of-band break messages using urgent data provided by the underlying protocol. Default is false.

Declaration

```
// C#  
public static bool DisableOOB { get; set; }
```

Property Type

System.Boolean

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

DrcpConnectionClass

This property specifies a logical name that identifies the DRCP connection pool that the ODP.NET connection will use. It will be used as a default if the DrcpConnectionClass property on the OracleConnection object is not set. It will be ignored for non-DRCP connections. Default value is null.

Declaration

```
// C#  
public static string DrcpConnectionClass { get; set; }
```

Property Type

System.String

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

HostnameDefaultServiceIsHost

This property specifies whether to default the service name to the hostname in Oracle Easy Connect Naming so that the service name does not need to be specified.

Declaration

```
// C#  
public static bool HostnameDefaultServiceIsHost { get; set; }
```

Property Type

System.Boolean

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

MapVectorColumnAsClob

This property specifies whether to retrieve vector types as native vectors or CLOB/VARCHAR2 in .NET.

Declaration

```
// C#  
public static bool MapVectorColumnAsClob { get; set; }
```

Property Type

System.Boolean

Remarks

When false (default), then native vector type for vector data is retrieved from the database. When true, then CLOB or VARCHAR2 for vector data is retrieved from the database.

The property allows easier migration from earlier ODP.NET versions that did not support a native vector data type to be upgraded with a simple code change and continue the existing behavior of using a CLOB or VARCHAR2 data type.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

NamesDirectoryPath

This property specifies the naming method order used for client name resolution lookups.

Declaration

```
// C#  
public static string NamesDirectoryPath { get; set; }
```

Property Type

System.String

Example

```
C#  
OracleConfiguration.NamesDirectoryPath = "(tnsnames, ldap)";
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

OciCompartment

This property gets or sets the Oracle Cloud Identifier (OCID) of the target database's compartment at the process level.

Declaration

```
// C#  
public static string OciCompartment { get; set; }
```

Property Type

System.String

Remarks

This property is optional by default. This property is mandatory if the `OciDatabase` property is set.

If `OciCompartment` is not set, then ODP.NET requests access to all the cloud tenancy's databases identified in the `OciTenancy` property.

There is no default value for this property.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

OciConfigurationFile

This property gets or sets the configuration file location at the process level where the OCI token authentication values can be found.

Declaration

```
// C#  
public static string OciConfigurationFile { get; set; }
```

Property Type

System.String

Remarks

This parameter is applicable to the OCI API key and OCI interactive authentication flows only and is optional.

The file names do not need to be specified since ODP.NET uses the standard file names.

If this parameter is not set, then ODP.NET uses the OCI config file in the default location: `C:/<user-profile>/.oci/config`. If the parameter is not set and the default configuration file is not found, then the user is prompted on the console to provide a region id for the OCI interactive authentication flow. A list of possible region ids will be displayed to the user. In the case of the OCI API key flow, the user will see an error.

On Windows, the Oracle `HOME` and `USERPROFILE` environment variables are used for the default directory location if they are set. Otherwise, Oracle `HOME` and `HOMEDRIVE` environment variables are used.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

OciDatabase

This property gets or sets the target database's Oracle Cloud Identifier (OCID) at the process level.

Declaration

```
// C#  
public static string OciCompartment { get; set; }
```

Property Type

System.String

Remarks

This property is optional.

If it is not set, then ODP.NET requests access to all the cloud compartment's databases identified in the `OciCompartment` property. If `OciCompartment` property is also not set, then ODP.NET requests access to all the tenancy's databases specified in the `OciTenancy` property.

There is no default value for this property.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

OciIamUrl

This property gets or sets the URL end point at the process level for getting the database token.

Declaration

```
// C#  
public static string OciIamUrl { get; set; }
```

Property Type

System.String

Remarks

This is a mandatory property for IAM alternate password use. If not set in that usage scenario, ODP.NET raises an error.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

OciProfile

This property gets or sets the OCI profile at the process level to use from the token authentication configuration file.

Declaration

```
// C#  
public static string OciProfile { get; set; }
```

Property Type

System.String

Remarks

This parameter is applicable to the OCI API key and OCI interactive authentication flows only and is optional.

ODP.NET uses the DEFAULT profile if no profile value is set.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

OciTenacy

This property gets or sets the Oracle Cloud Identifier (OCID) of the IAM user's cloud tenancy at the process level for retrieving the database token.

Declaration

```
// C#  
public static string OciTenacy { get; set; }
```

Property Type

System.String

Remarks

This is a mandatory property for IAM alternate password use. If not set in that usage scenario, ODP.NET raises an error.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

OracleDataSources

This property returns a collection of TNS entries added through OracleConfiguration.

Declaration

```
// C#  
public static OracleDataSourceCollection OracleDataSources
```

Property

Type:Oracle.ManagedDataAccess.Client.OracleDataSourceCollection

Returns a static OracleDataSourceCollection object.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)
- [OracleDataSourceCollection Class](#)

SuppressGetDecimalInvalidCastException

This property specifies whether to suppress the InvalidCastException and return a rounded-off 28 or 29 precision Oracle NUMBER value that can be represented as a .NET Decimal.

Declaration

```
// C#  
public static bool SuppressGetDecimalInvalidCastException {get; set;}
```

Property Type

System.Boolean

Remarks

Oracle NUMBER has a maximum of 38 precision. .NET Decimal has a maximum of 28 or 29 precision. When the `OracleDataReader.GetDecimal()`, `GetValue()`, or `OracleDataAdapter.Fill()` method is called for an Oracle NUMBER value that cannot be represented as a .NET Decimal, then ODP.NET throws an `InvalidCastException` because not all the precision can be preserved when converting the number to a .NET Decimal.

This behavior occurs when `SuppressGetDecimalInvalidCastException` is set to false, which is the default value.

When `SuppressGetDecimalInvalidCastException` is set to true, the resulting .NET Decimal is rounded off to 28 or 29 precision, allowing as much of the value to be represented without throwing an exception. If the resulting rounded number is larger than can be stored in a .NET Decimal, then an exception will be thrown, such as the number 1×10^{32} .

By using `OracleConfiguration.SuppressGetDecimalInvalidCastException`, this ODP.NET setting becomes enabled globally for the app. It can be overridden at the `OracleConnection`, `OracleDataReader`, or `OracleDataAdapter` level, which each has its own `SuppressGetDecimalInvalidCastException` property.

See Also

- `OracleConnection` [SuppressGetDecimalInvalidCastException](#)
- `OracleDataReader` [SuppressGetDecimalInvalidCastException](#)
- `OracleDataAdapter` [SuppressGetDecimalInvalidCastException](#)
- "[Oracle.DataAccess.Client](#) and [Oracle.ManagedDataAccess.Client](#) Namespaces"
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)
- [OracleDataSourceCollection Class](#)

TAFUsePooledConnections

This property indicates whether to use pre-existing connections from the pool for failover.

Declaration

```
// C#  
public static bool TAFUsePooledConnections
```

Remarks

By default, the property is set to true. An application can set this property to false to ensure ODP.NET always creates a new connection is created for failover.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

TcpConnectTimeout

This property specifies the time, in seconds, for a client to establish a TCP connection (PROTOCOL=tcp in the TNS connect address) to the database server before it can time out. Default value is 20 seconds.

Declaration

```
// C#  
public static string TcpConnectTimeout { get; set; }
```

Property Type

System.String

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

TimeZoneFileLocation

This property specifies the time zone file directory.

Declaration

```
// C#  
public string TimeZoneFileLocation {get; set;}
```

Property Type

System.String

Remarks

Countries and regions can alter when they observe daylight saving time or other changes for their time zones. Oracle publishes new time zone files to keep databases updated. Administrators then deploy these files to the database and clients to keep them in sync. Mismatched time zone settings between server and client can cause errors, such as ORA-01882: timezone region not found.

ODP.NET has built-in time zone settings that are up to date when it was published. To incorporate new settings, an administrator deploys the time zone file to a directory location and

sets `TimeZoneFileLocation` to that location before the ODP.NET app opens its first connection.

This property only accepts a directory location, not a file name.

ODP.NET will use the highest time zone file version among those in the directory and the ODP.NET embedded time zone file.

The backslash (\) is a special character in .NET that represents the beginning of an escape sequence. To specify a directory location, use any one of the following formats in .NET to correctly represent backslashes in a directory location:

```
// C#
OracleConfiguration.TimeZoneFileLocation= "D:\\oracle\\client\\timezone\\loc";
OracleConfiguration.TimeZoneFileLocation= @"D:\oracle\client\timezone\loc";
OracleConfiguration.TimeZoneFileLocation= "D:/oracle/client/timezone/loc";
```

Exceptions

`ArgumentException` - if a file is specified along with a directory.

`ArgumentException` - if an incorrect or empty directory is specified.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

TnsAdmin

This property specifies the `tnsnames.ora` and/or `sqlnet.ora` directory location.

Declaration

```
// C#
public static string TnsAdmin { get; set; }
```

Property Type

`System.String`

The backslash (\) is a special character in .NET that represents the beginning of an escape sequence. To specify a directory location, use any one of the following formats in .NET so that backslashes are correctly represented in a directory location:

```
// C#
OracleConfiguration.TnsAdmin = "D:\\oracle\\client\\admin";
OracleConfiguration.TnsAdmin = @"D:\oracle\client\admin";
OracleConfiguration.TnsAdmin = "D:/oracle/client/admin";
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

UseSNI

This property specifies whether to use Server Name Indication (SNI).

Declaration

```
// C#  
bool UseSNI { get; set;}
```

Property Type

System.Boolean

Remarks

Default is false. False disables using SNI.

When UseSNI is set to true, the following configuration rules must be obeyed. Any deviation will disable SNI automatically.

- Only the following CONNECT_DATA names can be used.
 - ServiceName
 - Server
 - InstanceName
 - ColocationTag
- The CONNECT_DATA name values can only contain alphanumeric, underscore, hyphen, or period characters.
- No more than two consecutive period characters
- No hyphen character before or after a period
- No hyphen at the beginning or end of the CONNECT_DATA name values above
- No period at the beginning or end of the CONNECT_DATA name values above

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)
- USE_SNI parameter in sqlnet.ora

OracleConfiguration Secure Connection Properties

OracleConfigurationSecure Connection properties are listed in [Table 6-37](#).

Table 6-37 OracleConfiguration Secure Connection Properties

Property	Description
AllowUncPaths	Allows or disallows ODP.NET access to configuration files with Universal Naming Convention location paths <i>Not available in ODP.NET, Unmanaged Driver</i>
OciCompartment	Gets or sets the Oracle Cloud Identifier (OCID) of the target database's compartment at the process level
OciDatabase	Gets or sets the target database's Oracle Cloud Identifier (OCID) at the process level
OciIamUrl	Gets or sets the URL end point at the process level for getting the database token
OciTenacy	Gets or sets the Oracle Cloud Identifier (OCID) of the IAM user's cloud tenancy at the process level for retrieving the database token
PasswordAuthentication	Sets the process level PASSWORD_AUTH parameter value
SEPSWalletLocation	Specifies the Secure External Password Store wallet location, typically used for PKCS #12 wallet files
SEPSWalletPassword	Stores the password to decrypt the PKCS #12 wallet located in the Secure External Password Store
SqlNetAllowWeakCrypto	Allows or disallows weaker encryption and checksum algorithms for database connections
SqlNetAllowedLogonVersionClient	Specifies the minimum authentication protocol that is to be used when authenticating against the Oracle database for a given application
SqlNetAuthenticationServices	Enables one or more authentication services, such as TCP/IP with SSL
SqlNetCloudUser	Specifies an user name for web server HTTP basic authentication
SqlNetCryptoChecksumClient	Specifies the checksum client behavior
SqlNetCryptoChecksumTypesClient	Specifies the crypto-checksum algorithms the client can use
SqlNetEncryptionClient	Specifies the encryption client behavior
SqlNetEncryptionTypesClient	Specifies encryption algorithms that the client can use
SqlNetIgnoreANOEncryptionForTCPS	Specifies whether to disable double encryption and double checksumming under certain conditions <i>Not available in ODP.NET, Unmanaged Driver</i>
SqlNetWalletOverride	Specifies whether the client overrides the strong authentication credential with the stored wallet password credential for the database connection
SqlNetURI	Specifies the WebSocket universal resource identifier
SSLCertificateThumbprint	Specifies the TLS/SSL certificate thumbprint in the certificate store for the ODP.NET connection to use
SSLServerDNMatch	Enforces the database server distinguished name (DN) matches its service name

Table 6-37 (Cont.) OracleConfiguration Secure Connection Properties

Property	Description
SSLVersion	Enforces the connection to use a specific SSL/TLS version <i>Not available in ODP.NET, Unmanaged Driver</i>
TLSExcludeVersion	Indicates which SSL/TLS versions connections should not be enabled for <i>Not available in ODP.NET, Unmanaged Driver</i>
TLSVersion	Enforces the connection to use a specific SSL/TLS version <i>Not available in ODP.NET, Unmanaged Driver</i>
TokenAuthentication	Sets the value for TOKEN_AUTH parameter at the process level
TokenLocation	Gets or sets the token location directory at the process level where the access token , or database token and private key files, are present for ODP.NET token authentication.
UseMITKerberos	Specifies whether to use MIT Kerberos or Kerberos.NET <i>Only available in ODP.NET, Managed Driver</i>
WalletLocation	Specifies the location of wallets
WalletPassword	Stores the password to decrypt the PKCS #12 wallet <i>Not available in ODP.NET, Unmanaged Driver</i>

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

AllowUncPaths

This property allows or disallows ODP.NET access to configuration files with Universal Naming Convention location paths.

Declaration

```
// C#
public static bool AllowUncPaths { get; set; }
```

Property Type

System.Boolean

Remarks

For ODP.NET to access configuration files, such as sqlnet.ora, on local network machines, users can specify Universal Naming Convention (UNC) paths.

The AllowUncPaths property enables or disables ODP.NET's ability to use these UNC paths.

The property default is true. When true, ODP.NET is allowed to access configuration files specified with UNC path locations.

When false, ODP.NET throws an ORA-50121 error when it tries to access any configuration file that uses an UNC path.

Server Message Block coercion attacks use UNC paths to have a target machine authenticate an attacker's computer. If successful, then the attacker copies that authentication to gain access to another system. Disabling UNC paths prevents this attack from being used.

An UNC patch consists of a double backslash, followed by a server name, host name, or IP address. Next is another backslash, followed by the share name. After that is a directory separator character with the directory name. This continues for additional subdirectories. An optional file name can be included at the end. Here's an example: `\MyHostName\MyShare\MyDirectory\MyFile.txt`

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

PasswordAuthentication

This property sets the process level PASSWORD_AUTH parameter value. The possible values are `OciToken` and `PasswordVerifier`. The default value is `PasswordVerifier`, which indicates the typical database username and password can be used for authentication.

Declaration

```
// C#  
public static OraclePasswordAuth PasswordAuthentication { get; set; }
```

Property Type

An `OraclePasswordAuth` enumeration

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

SEPSWalletLocation

This property specifies the Secure External Password Store wallet location, typically used for PKCS #12 wallet files.

Declaration

```
// C#  
public static string OracleConfiguration.SEPSWalletLocation { get; set; }
```

Property Type

System.String

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

SEPSWalletPassword

This property stores the password to decrypt the PKCS #12 wallet located in the Secure External Password Store.

Declaration

```
// C#  
public static OracleOpaqueString SEPSWalletPassword { get; set; }
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

SqlNetAllowedLogonVersionClient

This property specifies the minimum authentication protocol that is to be used when authenticating against the Oracle database for a given application.

Declaration

```
// C#  
public OracleAllowedLogonVersionClient SqlNetAllowedLogonVersionClient { set; }
```

Property Type

OracleAllowedLogonVersionClient

Values

An OracleAllowedLogonVersionClient enumeration value.

Example

```
// C#
OracleConfiguration.SqlNetAllowedLogonVersionClient =
OracleAllowedLogonVersionClient.Version12a;
```

Remarks

The default is `OracleAllowedLongVersionClient.12`.

📘 See Also

- *Oracle Database Net Services Reference* to better understand the minimum authentication protocol that is to be used when authenticating against the Oracle database for a given application.
- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)
- [OracleAllowedLogonVersionClient Enumeration](#)

SqlNetAllowWeakCrypto

This property allows or disallows weaker encryption and checksum algorithms for database connections.

Declaration

```
// C#
public static bool SqlNetAllowWeakCrypto { get; set; }
```

Property Type

`System.Boolean`

Remarks

When this parameter is `true`, ODP.NET can use weaker encryption and checksum algorithms to connect to Oracle database. When `false`, only stronger algorithms will be allowed to used for connections.

Default is `true`.

Encryption algorithms that ODP.NET supports and are considered weak:

- 3DES 112-bit
- 3DES 168-bit

No supported ODP.NET checksum algorithms are considered weak.

All other supported encryption and checksum algorithms are considered strong.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

SqlNetAuthenticationServices

This property enables one or more authentication services, such as TCP/IP with SSL.

Declaration

```
// C#  
public static string SqlNetAuthenticationServices { get; set; }
```

Property Type

System.String

Default value is NONE.

Valid Values: All, Kerberos5, NTS, TCPS, or NONE.

Note

ODP.NET Core does not currently support the Kerberos5 value.

Example

```
//C#  
con.SqlNetAuthenticationServices = "(nts, tcps)";
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

SqlNetCloudUser

This property specifies an user name for web server HTTP basic authentication.

Declaration

```
// C#  
public static string SqlNetCloudUser { get; set; }
```

Property Type

System.String

Default: none

Remarks

When you use a secure websocket protocol, the client uses this user as the user name for authentication. The password for this user should be stored in a wallet using mkstore utility commands.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

SqlNetCryptoChecksumClient

This property specifies the checksum client behavior.

Declaration

```
// C#  
public static string SqlNetCryptoChecksumClient { get; set; }
```

Property Type

System.String

Default value is accepted.

Valid Values:

- accepted - to enable the security service if required or requested by the database.
- rejected - to disable the security service, even if required by the database.
- requested - to enable the security service if the database allows it.
- required - to enable the security service and disallow the connection if the database is not enabled for the security service.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

SqlNetCryptoChecksumTypesClient

This property specifies the crypto-checksum algorithms the client can use.

Declaration

```
// C#  
public static string SqlNetCryptoChecksumTypesClient { get; set; }
```

Property Type

System.String

Valid Values:

- SHA1
- SHA256
- SHA384
- SHA512

Example

```
C#  
OracleConfiguration.SqlNetCryptoChecksumTypesClient = "(SHA1, SHA256, SHA512)";
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

SqlNetEncryptionClient

This property specifies the encryption client behavior.

Declaration

```
// C#  
public static string SqlNetEncryptionClient { get; set; }
```

Property Type

System.String

Default value is accepted.

Valid Values:

- accepted - to enable the security service if required or requested by the database.
- rejected - to disable the security service, even if required by the database.
- requested - to enable the security service if the database allows it.

- required - to enable the security service and disallow the connection if the database is not enabled for the security service.

📘 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

SqlNetEncryptionTypesClient

This property specifies encryption algorithms that the client can use.

Declaration

```
// C#  
public static string SqlNetEncryptionTypesClient { get; set; }
```

Property Type

System.String

Valid Values:

- AES128
- AES192
- AES256
- 3DES112
- 3DES168

Example

```
C#  
OracleConfiguration.SqlNetEncryptionTypesClient = "(aes128, aes192)";
```

📘 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

SqlNetIgnoreANOEncryptionForTCPS

This property specifies whether to disable double encryption and double checksumming under certain conditions.

Declaration

```
// C#  
public static bool SqlNetIgnoreANOEncryptionForTCPS { get; set; }
```

Property Type

System.Boolean

Remarks

When true and the connection uses TCPS (TLS), this property attempts to downgrade the Oracle Native Network Encryption (Advanced Networking Option) and checksumming from REQUIRED and REQUESTED to ACCEPTED. This setting indicates the intent to avoid double encryption and/or double data integrity when it's not desired. In those situations, ODP.NET will try to use TCPS only.

This property's default value is false. When false, ODP.NET will permit double encryption and double data integrity if they are enabled.

The parameter is ignored in either of these two scenarios:

- Transport is not TCPS
- NNE (ANO) encryption and/or checksumming level is not set to REQUIRED or REQUESTED

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

SqlNetWalletOverride

This property specifies whether the client overrides the strong authentication credential with the stored wallet password credential for the database connection.

Declaration

```
// C#  
public static bool SqlNetWalletOverride { get; set; }
```

Property Type

System.Boolean

Description

Default is false. By setting this value to true, ODP.NET uses the stored wallet password credential to connect.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

SqlNetURI

This property specifies the WebSocket universal resource identifier.

Declaration

```
// C#  
public static string SqlNetURI { get; set; }
```

Property Type

System.String

This property is Request-URI of the HTTP GET method. It identifies the endpoint of the WebSocket connection, both to allow multiple domains to be served from one IP address and to allow multiple WebSocket endpoints to be served by a single server.

When setting the URI in unmanaged ODP.NET, it cannot begin with a forward slash character. For managed ODP.NET and ODP.NET Core, the forward slash at the beginning of the URI is optional.

Default value is sqlnet.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

SSLCertificateThumbprint

This property specifies the TLS/SSL certificate thumbprint in the certificate store for the ODP.NET connection to use.

Declaration

```
// C#  
public static string SSLCertificateThumbprint { get; set; }
```

Remarks

When there are multiple certificates possible for ODP.NET to use from the certificate store, specifying the thumbprint tells ODP.NET which certificate to connect the user with. This avoids the user from performing certificate selection manually, such as via a graphical interface. Users can select a certificate manually using the `OracleConnection.AllowCertificateSelectionUI` property.

ODP.NET supports certificates through thumbprint only. It supports the SHA1 and SHA2 algorithms.

ODP.NET expects thumbprints in the format of <Algorithm>:<Hash>. For example,

- SHA1:1B:11:01:5E:D1:7C:20:B2:62:39:3E:04:7B:83:47:DE:70:2E:4E:44
- SHA256:CD:C6:8B:0D:36:FE:20:0E:7F:86:5F:94:0B:4D:EC:EE:1B:C0:D1:09

Or

- SHA1:1B11015ED17C20B262393E047B8347DE702E4E44
- SHA256:CDC68B0D36FE200E7F865F940B4DECEE1BC0D109

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

SSLServerDNMatch

This property enforces the database server distinguished name (DN) matches its service name. Default value is false.

Declaration

```
// C#  
public static bool SSLServerDNMatch { get; set; }
```

Property Type

System.Boolean

Valid Values:

- true - to enforce a match. If the DN matches the service name, then the connection succeeds. If the DN does not match the service name, then the connection fails.
- false - to not enforce a match. If the DN does not match the service name, then the connection is successful, but an error is logged to the sqlnet.log file.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

SSLVersion

This property enforces the connection to use a specific SSL/TLS version.

Declaration

```
// C#  
public static string SSLVersion { get; set; }
```

Property Type

System.String

Description

Supported values are: undetermined |1.2|TLSv1.2|1.3|TLSv1.3. To specify more than one SSL/TLS version, enclose the version numbers between parentheses and separate the values with a comma. For example, (1.2,1.3).

Starting with ODP.NET 23.9, you can specify a minimum TLS version by appending the plus (+) sign after the version, such as 1.2+ or TLSv1.2+. This value indicates to use TLS 1.2 or higher, whichever highest version the database server supports.

In some cases, apps may want to set a minimum TLS version, but not support one or more higher versions. To identify excluded versions, set the TLSExcludeVersion property on OracleConfiguration, OracleConnection, or equivalent.

The default value is an undetermined version. An undetermined version means ODP.NET is allowed to use any TLS version, this property is allowed to specify. An exception will be thrown for invalid value.

This property overrides the SSL_VERSION parameter.

Example

```
OracleConfiguration.SSLVersion = 1.2+;
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

TLSExcludeVersion

This property indicates which SSL/TLS versions connections should not be enabled for.

Declaration

```
// C#  
public static string TLSExcludeVersion { get; set; }
```

Property Type

System.String

Description

The `TLSVersion` or `SSLVersion` properties sets the supported connection SSL/TLS versions. `TLSExcludeVersion` sets the versions that the app connections cannot use.

Typically, `TLSExcludeVersion` is used when the TLS/SSL version has a minimum value set, such as `TLSv1.2+`, and the app wants to prevent connections from using a version above the minimum value, such as `TLS 1.3`.

Supported exclusion values are: `1.2|TLSv1.2|1.3|TLSv1.3`. To specify more than one SSL/TLS version to exclude, enclose the version numbers between parentheses and separate the values with a comma. For example, `(1.2,1.3)`.

This property has no default value.

An exception will be thrown for invalid value.

This property overrides the `TLS_EXCLUDE_VERSION` parameter, including the full connect descriptor and Easy Connect Plus configuration.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

TLSVersion

This property enforces the connection to use a specific SSL/TLS version.

Declaration

```
// C#  
public static string TLSVersion { get; set; }
```

Property Type

`System.String`

Description

Supported values are: `undetermined|1.2|TLSv1.2|1.3|TLSv1.3`. To specify more than one SSL/TLS version, enclose the version numbers between parentheses and separate the values with a comma. For example, `(1.2,1.3)`.

Starting with ODP.NET 23.9, you can specify a minimum TLS version by appending the plus (+) sign after the version, such as `1.2+` or `TLSv1.2+`. This value indicates to use TLS 1.2 or higher, whichever highest version the database server supports.

In some cases, apps may want to set a minimum TLS version, but not support one or more higher versions. To identify excluded versions, set the `TLSExcludeVersion` property on `OracleConfiguration`, `OracleConnection`, or equivalent.

The default value is an undetermined version. An undetermined version means ODP.NET is allowed to use any TLS version, this property is allowed to specify. An exception will be thrown for invalid value.

This property overrides the SSL_VERSION parameter.

Example

```
OracleConfiguration.TLSVersion = 1.2+;
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

TokenAuthentication

This property sets the value for TOKEN_AUTH parameter at the process level. The possible values for this property include OracleTokenAuth.OciToken, OracleTokenAuth.Disabled, OracleTokenAuth.OAuth and so on.

Declaration

```
// C#
public static OracleTokenAuth TokenAuthentication { get; set; }
```

Table 6-38 OracleTokenAuth Members

Member Name	Description
Disabled	Default value. Token authentication is DISABLED.
OAuth	Token authentication enabled for Microsoft Entra ID.
OciApiKey	Token authentication using OCI_API_KEY authentication flow. <i>Not available in ODP.NET, Unmanaged Driver</i>
OciInstancePrincipal	Token authentication using OCI_INSTANCE_PRINCIPAL authentication flow. <i>Not available in ODP.NET, Unmanaged Driver</i>
OciDelegationToken	Token authentication using OCI_DELEGATION_TOKEN authentication flow. <i>Not available in ODP.NET, Unmanaged Driver</i>
OciInteractive	Token authentication using OCI_INTERACTIVE authentication flow. <i>Not available in ODP.NET, Unmanaged Driver</i>
OciDefault	Token authentication using the most appropriate method depending on the application environment. ODP.NET first attempts to retrieve the token using the OCI API key. If that fails, then ODP.NET checks whether it is in a Cloud Shell environment to attempt to get the token using delegation token authentication flow. Lastly, it attempts to use the instance principal authentication flow. <i>Not available in ODP.NET, Unmanaged Driver</i>

Table 6-38 (Cont.) OracleTokenAuth Members

Member Name	Description
OciToken	Token authentication is enabled for Oracle Identity and Access Management.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

TokenLocation

This property gets or sets the token location directory at the process level where the access token, or database token and private key files, are present for ODP.NET token authentication. The file names do not need to be specified.

Declaration

```
// C#  
public static string TokenLocation { get; set; }
```

Property Value

A directory location.

Exceptions

None.

Description

This property is mandatory for OAuth file-based access tokens, but optional for database tokens.

In the access token case, this property can be the directory where the file “token” is, or the full-path specification of the token file.

For OCI IAM PoP tokens, a file’s full-path specification can be specified if it does not have the standard name, “token”. A file name does not need to be specified when it uses this standard, fixed name. Bearer token default token location is an empty string.

Note

Only managed ODP.NET and ODP.NET Core can use this property.

Remarks

Applications can override the default location of signature token and private key by setting a new custom location through this property.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

UseMITKerberos

This property specifies whether to use MIT Kerberos or Kerberos.NET.

Declaration

```
// C#  
public static bool UseMITKerberos { get; set; }
```

Property Type

System.Boolean

Remarks

This property only applies to managed ODP.NET as it can use either MIT Kerberos or Kerberos.NET.

When set to true (default), managed ODP.NET will use MIT Kerberos. MIT Kerberos must be installed separately.

When set to false, it uses the Oracle.ManagedDataAccess.Kerberos NuGet package and Kerberos.NET. Oracle recommends using Kerberos.NET.

The option to use managed ODP.NET with Kerberos.NET became available in ODP.NET 23.9.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

WalletLocation

This property specifies the location of wallets. Wallets are certificates, keys, and trustpoints processed by SSL/TLS.

Declaration

```
// C#  
public static string WalletLocation { get; set; }
```

Property Type

System.String

Remarks

Microsoft Certificate Store (MCS) and file system wallets are supported.

This property can accept file path or URL values that contain the wallet file.

It supports single sign-on files (.sso file extension) and PKCS #12 files (.p12 file extension). Since wallet location is a directory location, not file location, ODP.NET chooses the file to use based on the criteria:

- When the wallet password is not supplied:
 - Reads the .sso file, if present
 - Reads the .p12 file, if present
- When the wallet password is supplied:
 - Reads the .p12 file, if present

Note

MCS wallets are supported only on Windows operating systems.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

WalletPassword

This property stores the password to decrypt the PKCS #12 wallet.

Declaration

```
// C#  
public static OracleOpaqueString WalletPassword {get; set;}
```

Property Value

The password to decrypt the encrypted PKCS #12 wallet.

Remarks

Wallet passwords support the same functionality as traditional ODP.NET passwords.

Since PKCS #12 files are encrypted, apps must provide the wallet password as an `OracleOpaqueString`. When the wallet password is provided through the `OracleConfiguration.WalletPassword` or the `OracleConnection.WalletPassword` class property, ODP.NET loads the associated PKCS #12 file as designated by the wallet location. Alternately, if no password is provided, then ODP.NET loads the associated SSO file.

Changing the `WalletPassword` for an existing connection pool does not create a new pool. Only setting the property or unsetting it creates a new pool.

Note

The `OracleConfiguration.WalletPassword` is associated with the `OracleConfiguration.WalletLocation` and the `OracleConnection.WalletPassword` is associated with the `OracleConnection.WalletLocation`.

Example

```
//Create OracleOpaqueString for wallet password
string walletPassword = "<WALLET PASSWORD>";
OracleOpaqueString opaPwd = new OracleOpaqueString();
foreach (char c in walletPassword)
    opaPwd.AppendChar(c);
opaPwd.MakeReadOnly();

//Create and open connection
OracleConnection conn = new OracleConnection("user id=HR; password=<PASSWORD>");
conn.WalletLocation = "C:\\wallet\\location\\";
conn.WalletPassword = opaPwd;
conn.Open();
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

OracleConfiguration Command Properties

OracleConfiguration Command properties are listed in [Table 6-39](#).

Table 6-39 OracleConfiguration Command Properties

Property	Description
AddOracleTypesDeserialization	Adds ODP.NET-specific data types to the "allow" list to permit deserialization into <code>DataSet</code> or <code>DataTable</code>
BindByName	Specifies whether the binding method used for the parameter collection is by name or by position
CommandTimeout	Specifies the number of seconds the command is allowed to execute before terminating the execution with an exception
GetDecimalRetainTrailingZeros	Specifies whether to retain trailing zeros from an Oracle <code>NUMBER</code> in a .NET <code>Decimal</code>

Table 6-39 (Cont.) OracleConfiguration Command Properties

Property	Description
MapVectorColumnAsClob	Specifies whether to retrieve vector types as native vectors or CLOB/VARCHAR2 in .NET <i>Not available in ODP.NET, Unmanaged Driver</i>
ProviderTypeNLSSensitiveToString	Specifies whether to honor the data type's NLS format when the ToString() method is called <i>Not available in ODP.NET, Unmanaged Driver</i>
UseClientInitiatedCQN	Specifies whether to use Client Initiated Continuous Query Notification (CICQN) or traditional Continuous Query Notification (CQN) that does not rely on a persistent connection

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

AddOracleTypesDeserialization

This static method adds ODP.NET-specific data types to the “allow” list to permit deserialization into DataSet or DataTable.

Declaration

```
// C#
public static void AddOracleTypesDeserialization();
```

Remarks

In newer .NET versions, the allowed DataSet and DataTable data types that can be deserialized are now restricted for security reasons. This change applies to .NET 5, plus .NET Core and .NET Framework updates. If application DataSets and DataTables use Oracle data types with one of these new .NET versions, use the AddOracleTypesDeserialization method to add ODP.NET-specific data types to the “allow” list so that they can be deserialized. If an attempt is made to deserialize ODP.NET-specific types without adding them to the “allow” list, an ODP.NET type initializer exception will be encountered.

If other software that is part of the application adds data types to the allow list as well, be careful not to overwrite the ODP.NET allowed types. Be sure to only append to the allow list, not overwrite it. If overwriting does occur, call AddOracleTypesDeserialization method afterwards to add the ODP.NET types back to the list.

The ODP.NET AddOracleTypesDeserialization method call itself appends to the allow list. It does not overwrite existing entries.

This property is available in ODP.NET 19.10 and higher. It has also been backported to the latest ODP.NET 18c, 12.2, and 12.1 patches.

Example

```
// C#
// Sample demonstrating loading an XML file with Oracle data types into DataSet

using System.Data;
using Oracle.ManagedDataAccess.Client;

class OracleTypeDeserializationSample
{
    static void Main()
    {
        OracleConfiguration.AddOracleTypesDeserialization();
        DataSet ds = new DataSet();
        ds.ReadXml("dsch1.xml");
    }
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

BindByName

This property specifies whether the binding method used for the parameter collection is by name or by position. Default value (false) is bind by position.

Declaration

```
// C#
public static bool BindByName { get; set; }
```

Property Type

System.Boolean

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

CommandTimeout

This property specifies the number of seconds the command is allowed to execute before terminating the execution with an exception. Default value is 0 seconds, which results in no time limit.

Declaration

```
// C#  
public static int CommandTimeout { get; set; }
```

Property Type

System.Int32.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

GetDecimalRetainTrailingZeros

This property specifies whether to retain trailing zeros from an Oracle NUMBER in a .NET Decimal.

Declaration

```
// C#  
public static bool GetDecimalRetainTrailingZeros { get; set; }
```

Property Type

System.Boolean

Remarks

Starting with ODP.NET 19.11, Oracle NUMBER column values retrieved as .NET Decimals retain a trailing zero if the number of digits on the right hand side of the decimal point is odd in number when this property is set to true.

The default value is false.

This property can be used whenever retrieving data into a .NET Decimal, including the following scenarios:

- `OracleDataReader.GetDecimal();`
- `OracleDecimal.Value`
- Parameter output value of type `Decimal`
- JSON data with `Decimal` in it
- UDT with `Decimal` attribute

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

ProviderTypeNLSSensitiveToString

This property specifies whether to honor the data type's NLS format when the ToString() method is called.

Declaration

```
// C#  
public static bool ProviderTypeNLSSensitiveToString { get; set; }
```

Property Value

System.Boolean

Remarks

Default value is true. When true, ODP.NET honors the database session NLS format when ToString() is called for the provider type.

Prior to version 23.6, ODP.NET returned data in a default format that ignored the database session's NLS format. If you wish to use this earlier behavior, set the property to false.

This property applies to the following provider types: OracleDate, OracleDecimal, OracleTimeStamp, OracleTimeStampLTZ, and OracleTimeStampTZ.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)
- [OracleDataSourceCollection Class](#)

UseClientInitiatedCQN

This property specifies whether to use Client Initiated Continuous Query Notification (CICQN) or traditional Continuous Query Notification (CQN) that does not rely on a persistent connection.

Declaration

```
// C#  
public static bool UseClientInitiatedCQN {get; set;}
```

Remarks

When set to true, ODP.NET creates and uses a separate connection to receive server-initiated database change notifications. This is known as CICQN. This single connection aggregates all notifications for the pool, but the connection does not count toward the Min Pool Size nor the Max Pool Size limits. For managed ODP.NET, if there are five connection pools using CICQN, then ODP.NET will then have five additional connections, one per pool.

If false (default), then ODP.NET will use traditional CQN to receive the database change notifications. The client creates a listening end point (i.e. IP address and port) that does not rely on a database connection to continuously exist.

CICQN is commonly used in cloud scenarios or if firewalls between the database and client do not allow access to the client port. Otherwise, traditional CQN is used.

ODP.NET CICQN requires Oracle Database 19c or higher.

UseClientInitiatedCQN must be set to its intended value prior to opening a database connection.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

OracleConfiguration Directories Properties

OracleConfiguration Directories properties are listed in [Table 6-40](#) .

Table 6-40 OracleConfiguration Directories Properties

Property	Description
DefaultAdminContext	Specifies the default directory entry that contains a LDAP Oracle Context from which connect identifiers can be created, modified, or looked up
DirectoryServers	Lists the host names and port numbers of the primary and alternate LDAP directory servers
DirectoryServerType	Specifies the directory server type used, such as Oracle Internet Directory or Microsoft Active Directory
LdapAdmin	Specifies the ldap.ora directory location
NamesLdapAuthenticateBind	Specifies whether the LDAP naming adapter should attempt to authenticate using a specified wallet when it connects to the LDAP directory to resolve the name in the connect string
NamesLdapAuthenticateBindMethod	Specifies the authentication method the client LDAP naming adapter should use while connecting to the LDAP directory to resolve connect string names
NamesLdapConnTimeout	specifies the non-blocking connection timeout to the LDAP server

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

DefaultAdminContext

This property specifies the default directory entry that contains a LDAP Oracle Context from which connect identifiers can be created, modified, or looked up.

Declaration

```
// C#  
public static string DefaultAdminContext { get; set; }
```

Property Type

System.String

Values

Valid distinguished name

Example

```
// C#  
OracleConfiguration.DefaultAdminContext=@"o=OracleSoftware,c=US";
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

DirectoryServers

This property lists the host names and port numbers of the primary and alternate LDAP directory servers.

Declaration

```
// C#  
public static string DirectoryServers { get; set; }
```

Property Type

System.String

Use the following format to list the host names and port numbers:

<HOST>:<PORT>[:<TLS/SSL PORT>]

Example

```
// C#
OracleConfiguration.DirectoryServers="(ldap-server1:389, ldap-server2:400:636)";
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

DirectoryServerType

This property specifies the directory server type used, such as Oracle Internet Directory or Microsoft Active Directory.

Declaration

```
// C#
public static string DirectoryServerType { get; set; }
```

Property Type

System.String

Values

- *oid* for Oracle Internet Directory
- *ad* for Microsoft Active Directory

Example

```
// C#
OracleConfiguration.DirectoryServerType=@"oid";
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

LdapAdmin

This property specifies the `ldap.ora` directory location.

Declaration

```
// C#
public static string LdapAdmin { get; set; }
```

Property Type

System.String

The LdapAdmin setting works in conjunction with the TnsAdmin setting to determine the ldap.ora search order.

Example

```
// C#
OracleConfiguration.LdapAdmin=@"D:\user\ldap\";
```

Remarks

ODP.NET will search for ldap.ora using the location search order as listed in [Oracle Configuration Files](#) for ODP.NET Core and in [Oracle Client Configuration File Settings](#) for managed ODP.NET.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

NamesLdapAuthenticateBind

This property specifies whether the LDAP naming adapter should attempt to authenticate using a specified wallet when it connects to the LDAP directory to resolve the name in the connect string.

Declaration

```
// C#
public static bool NamesLdapAuthenticateBind { get; set; }
```

Property Type

System.Boolean

If the parameter is set to true, then the LDAP connection is authenticated using a wallet whose location must be specified in the OracleConfiguration.WalletLocation property.

If the parameter is set to false, then the LDAP connection is established using an anonymous bind.

Default value is false.

Example

```
// C#
OracleConfiguration.NamesLdapAuthenticateBind = true;
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

NamesLdapAuthenticateBindMethod

This property specifies the authentication method the client LDAP naming adapter should use while connecting to the LDAP directory to resolve connect string names.

Declaration

```
// C#  
public static string NamesLdapAuthenticateBindMethod { get; set; }
```

Property Type

System.String

Remarks

The simple authentication method over LDAPS (LDAP over TLS connection) is supported.

Store the directory entry distinguished name and password in an Oracle wallet. When the client connects to the LDAP server, it is authenticated using the credentials stored in this wallet. The wallet must contain client certificates. Its trust store must contain the certificates issued by the certificate authority of the LDAP server.

The parameter value is a string (ldaps_simple_auth).

The LDAP naming adapter uses the `oracle.ldap.client.dn` and `oracle.ldap.client.password` entries from the wallet for authenticating to the LDAP server. If these entries are not present, then the client attempts an anonymous authentication using TLS or LDAPS.

Default

none

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

NamesLdapConnTimeout

This property specifies the non-blocking connection timeout to the LDAP server.

Declaration

```
// C#
public static Int32 NamesLdapConnTimeout { get; set; }
```

Property Type

System.Int32

This property sets the time to wait (in seconds) for a LDAP server connection before a time out error can occur.

The minimum value is -1, which indicates no connection timeout. The default value is 15 seconds.

Example

```
// C#
OracleConfiguration.NamesLdapConnTimeout = -1;
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

OracleConfiguration High Availability Properties

OracleConfiguration High Availability properties are listed in [Table 6-41](#).

Table 6-41 OracleConfiguration High Availability Properties

Property	Description
ChunkMigrationConnectionTimeout	This setting pauses the connection request until the migration completes or fails to migrate in the specified time, then connect to the correct shard with the data requested. <i>Not Available in ODP.NET, Unmanaged Driver</i>
DbNotificationAddress	Specifies the address that the provider listens to for all notifications sent by the database for continuous query notification feature
DbNotificationPort	Specifies the port number that the provider listens to for all notifications sent by the database for continuous query notification, Fast Connection Failover, and/or Runtime Connection Load Balancing features
DistributedTransactionRecoveryAddress	Specifies the machine name or IP address the OraMTS recovery service runs on to resolve database in-doubt transactions
DistributedTransactionRecoveryPort	Specifies the port the OraMTS recovery service listens on to resolve database in-doubt transactions
DistributedTransactionTimeToLive	Specifies the time the transaction can remain inactive after it has been detached or delisted from the database

Table 6-41 (Cont.) OracleConfiguration High Availability Properties

Property	Description
HAEvents	Enables the application to receive Fast Connection Failover events for maintaining application high availability
MaxCallsPerRequestBoundary	Specifies the maximum request size, or ODP.NET calls, beyond which Application Continuity replay will be disabled
OnsConfigFile	Specifies the configuration file to define Oracle Notification Service (ONS) behavior
OnsMode	Specifies the ONS daemon mode as either local or remote
OnsProtocol	Defines whether the ONS connection uses TCP/IP or TCP/IP with SSL/TLS
OnsWalletLocation	Specifies the directory location of the ONS wallets
OracleOnsServers	Returns a collection of logical ONS servers added through OracleConfiguration
ServiceRelocationConnectionTimeout	Specifies the time to wait before retrying connecting to a service that becomes unavailable.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

ChunkMigrationConnectionTimeout

In sharding, a connection request to a specific shard can occur while data is migrating from the original shard to a new one. This setting will pause the connection request until the migration completes or fails to migrate in the specified time, then connect to the correct shard with the data requested.

Declaration

```
// C#
public static int ChunkMigrationConnectionTimeout {get;set;}
```

Property Value

The maximum time, in seconds, for a pooled connection request to pause waiting for a chunk migration or split partitionset for a particular chunk.

Remarks

If a specific shard connection request occurs while data is migrating to a new shard or being split and moved to a new shard through a split partitionset operation, then this setting pauses the connection request until the data migration completes or fails in the specified time. It then connects to the correct shard with the data requested.

Default is 120 seconds. This setting requires connection pooling to be enabled.

ODP.NET reads the chunk migration connection timeout setting only upon pool creation. A new timeout setting will not be used until a new pool is created.

This setting is in effect if

- write access is required and a chunk is read-only, or
- during a chunk migration or split partitionset, the chunk becomes unavailable to read.

If the connection requires only read-only access, then this setting will not be used unless the chunk becomes unavailable.

If the connection requires write access and the chunk does not become writable or available before the chunk migration connection timeout expires, then a new connection will be created. If the connection cannot be created, then an `OracleException` will occur.

During chunk migration or split partitionset, an ODP.NET connection request obeys only the chunk migration connection timeout. All other connection timeout values, such as `OracleConnection.ConnectionTimeout`, are ignored.

Oracle recommends setting the ODP.NET chunk migration connection timeout to a value greater than the chunk move server timeout. One way to set this server timeout is using the Global Data Services Control Utility (GDSCtl) `move chunk` command's timeout option. This server-side timeout only applies to chunk migrations, not split partitionset.

The `OracleConnection ChunkMigrationConnectionTimeout` property inherits from the `OracleConfiguration ChunkMigrationConnectionTimeout` property. The `OracleConnection` property can be separately set to override the property set in the `OracleConfiguration` class.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

DbNotificationAddress

This property specifies the address that the provider listens to for all notifications sent by the database for continuous query notification feature.

Declaration

```
// C#  
public static string DbNotificationAddress { get; set; }
```

Property Type

`System.String`

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

DbNotificationPort

This property specifies the port number that the provider listens to for all notifications sent by the database for continuous query notification, Fast Connection Failover, and/or Runtime Connection Load Balancing features. A value of -1 allows directs the provider to use a random port.

Declaration

```
// C#  
public static int DbNotificationPort { get; set; }
```

Property Type

System.Int32.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

DistributedTransactionRecoveryAddress

This property specifies the machine name or IP address the OraMTS recovery service runs on to resolve database in-doubt transactions.

Declaration

```
// C#  
public static string DistributedTransactionRecoveryAddress { get; set; }
```

Property Type

System.String

Remarks

The default value is the local machine name.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

DistributedTransactionRecoveryPort

This property specifies the port the OraMTS recovery service listens on to resolve database in-doubt transactions.

Declaration

```
// C#  
public static int DistributedTransactionRecoveryPort { get; set; }
```

Property Type

System.Int32

Remarks

The default value is 2030.

The port must be between 1 and 65535, or the default will be used.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

DistributedTransactionTimeToLive

This property specifies the time the transaction can remain inactive after it has been detached or delisted from the database.

Declaration

```
// C#  
public static int DistributedTransactionTimeToLive { get; set; }
```

Property Type

System.Int32

Property Value

The unit of time is seconds.

Remarks

The default value is 120 seconds.

The time to live must be between 1 and `int.MaxValue`, or the default will be used.

Once this time expires, the transaction is automatically terminated by the provider.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

HAEvents

This property enables the application to receive Fast Connection Failover events for maintaining application high availability. Default is true.

Declaration

```
// C#  
public static bool HAEvents { get; set; }
```

Property Type

`System.Boolean`

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

MaxCallsPerRequestBoundary

This property specifies the maximum request size, or ODP.NET calls, beyond which Application Continuity replay will be disabled.

Declaration

```
// C#  
public static int MaxCallsPerRequestBoundary { get; set; }
```

Property Type

`System.Int32`

Remarks

In Application Continuity, ODP.NET stores all the driver calls that occur between request boundaries, generally when connections are checked out and in. Having this call record is what makes session restoration possible after a failover. For long-running connections, more calls to restore mean more memory is necessary to retain all the changes between request boundaries. ODP.NET allows you to disable session replay when this memory usage becomes too high by setting a maximum number of calls to retain between request boundaries.

If the property value is set to 100, replay will be disabled for any request with more than 100 calls. Disabling replay applies only to the connection request that exceeds the limit. Other connection requests below the limit remain enabled and can be recovered after a failover. Replay will be re-enabled at the next BeginRequest call on the same connection.

By default, the property value is `Int32.MaxValue`.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

OnsConfigFile

This property specifies the configuration file to define Oracle Notification Service (ONS) behavior. The file specified should contain the same local port and remote port values as specified in the `ons.config` file used by the local ONS daemon. This will enable the application to receive events from the local ONS daemon.

Declaration

```
// C#  
public static string OnsConfigFile { get; set; }
```

Property Type

`System.String`

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

OnsMode

This property specifies the ONS daemon mode as either "local" or "remote".

For local configuration, configure and make available ONS on the node where ODP.NET is running so that ODP.NET can receive events directly from the local ONS daemon.

Remote configuration is used when the application directly receives ONS events from the ONS daemons running on remote machines. One of the advantages of this configuration is that no ONS daemon is needed on the client end and; therefore, there is no need to manage this process.

Declaration

```
// C#  
public static OnsConfigMode OnsMode { get; set; }
```

Property Type

System.String

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

OnsProtocol

This property defines whether the ONS connection uses TCP/IP or TCP/IP with SSL/TLS.

Declaration

```
// C#  
public static string OnsProtocol { get; set; }
```

Property Type

System.String

Usage Notes

Valid values are TCP or TCPS. Default value is TCP.

Example

```
OracleConfiguration.OnsProtocol=TCPS
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

OnsWalletLocation

This property specifies the directory location of the ONS wallets. Wallets are certificates, keys, and trustpoints processed by SSL/TLS.

Declaration

```
// C#  
public static string OnsWalletLocation { get; set; }
```

Property Type

System.String

Usage Notes

Default value is none.

Example

```
OracleConfiguration.OnsWalletLocation=@"D:\user\wallets\"
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

OracleOnsServers

This property returns a collection of logical ONS servers added through OracleConfiguration.

Declaration

```
// C#  
public static OracleOnsServerCollection OracleOnsServers
```

Property

Type:Oracle.ManagedDataAccess.Client.OracleOnsServerCollection

Returns a static OracleOnsServerCollection object.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)
- [OracleDataSourceCollection Class](#)

ServiceRelocationConnectionTimeout

This property specifies the time to wait before retrying connecting to a service that becomes unavailable. Default value is 90 seconds. Whenever a database service becomes unavailable, such as due to a service being relocated, an application can encounter numerous connectivity errors during this time. To avoid unnecessary connection attempts to an unavailable service which will result in an error, the driver will block any connection attempts until the service is up or until this property's specified time limit expires from the time when the service DOWN event was received, whichever comes first.

Declaration

```
// C#
public static string ServiceRelocationConnectionTimeout { get; set; }
```

Property Type

System.String

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

OracleConfiguration Performance Properties

OracleConfiguration Performance properties are listed in [Table 6-42](#).

Table 6-42 OracleConfiguration Performance Properties

Property	Description
DefaultSDUSize	Specifies the session data unit size (SDU) for Oracle networking communication between client and server
FetchSize	Specifies the total memory size, in bytes, that the provider allocates to cache data fetched in one database round-trip
LoadBalancing	Enables the application to receive runtime connection load balancing information
MaxStatementCacheSize	Specifies the maximum number of statements that can be cached when self-tuning is enabled
Pipelining	Enables asynchronous execution on the database server side
PerformanceCounters	Specifies the ODP.NET connection performance counters to publish so that they can be monitored
ReceiveBufferSize	Specifies the buffer space limit for receive operations of sessions

Table 6-42 (Cont.) OracleConfiguration Performance Properties

Property	Description
RowsToFetchPerRoundTrip	Specifies the total number of rows to retrieve per database round trip <i>Not Available in ODP.NET, Unmanaged Driver</i>
SelfTuning	Specifies whether self-tuning is enabled for an ODP.NET application
SendBufferSize	Specifies the buffer space limit for send operations of sessions
StatementCacheSize	Specifies the number of cursors or statements to be cached for each database connection
TcpNoDelay	Preempts delays in buffer flushing within the TCP/IP protocol stack

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

DefaultSDUSize

This property specifies the session data unit size (SDU) for Oracle networking communication between client and server.

Declaration

```
// C#
public static int DefaultSDUSize { get; set; }
```

Remarks

Before sending data across the network, ODP.NET or Oracle database buffers and encapsulates data into the SDU. They send the data stored in this buffer when the buffer is full, flushed, or when database server tries to read data. When large amounts of data are being transmitted or when the message size is consistent, adjusting the size of the SDU buffers can improve performance, network utilization, or memory consumption.

The default value is 65536 bytes, equivalent to 64 KB.

ODP.NET and Oracle database can each set its own SDU size. The actual SDU size used is negotiated at connect time. The smaller of the two values is chosen. Oracle AI Database 26ai uses a 64 KB SDU size. Earlier database releases set their default SDU size to 8 KB.

The OracleConfiguration SDU size can only be set before the first connection is established globally in the the .NET domain.

The OracleConnection SDU size can only be set before the connection is established. Moreover, the OracleConnection SDU size for a connection pool can only be set one time. All follow on set attempts are ignored for that pool.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

FetchSize

This property specifies the total memory size, in bytes, that the provider allocates to cache data fetched in one database round-trip. Default value is 131072.

Declaration

```
// C#  
public static int FetchSize { get; set; }
```

Property Type

System.Int32

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

LoadBalancing

This property enables the application to receive runtime connection load balancing information. Default is true.

Declaration

```
// C#  
public static bool LoadBalancing { get; set; }
```

Property Type

System.Boolean

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

MaxStatementCacheSize

This property specifies the maximum number of statements that can be cached when self-tuning is enabled. Default value is 100.

Declaration

```
// C#  
public static int MaxStatementCacheSize { get; set; }
```

Property Type

System.Int32

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

PerformanceCounters

This property specifies the ODP.NET connection performance counters to publish so that they can be monitored.

Declaration

```
// C#  
public static Int64 PerformanceCounters { get; set; }
```

Property Type

System.Int64

Remarks

ODP.NET enables monitoring many different connection counters, including pooled and non-pooled connections. These counters can be monitored individually or together. Developers can set which counters to monitor prior to application startup using the PerformanceCounters property.

By default, performance counters are not enabled.

① See Also

- [Connection Performance Counters](#) for more information on setting up and configuring counters.
- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

Pipelining

This property enables asynchronous execution on the database server side.

Declaration

```
// C#  
public static bool Pipelining { get; set; }
```

Property Type

System.Boolean

Description

Default is false.

To enable pipelining, that is, database server asynchronous execution; set this property to true, and enable asynchronous ODP.NET, that is, client side asynchronous execution as well. Pipelining can only be enabled if async ODP.NET is enabled.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

ReceiveBufferSize

This property specifies the buffer space limit for receive operations of sessions.

Declaration

```
// C#  
public static int ReceiveBufferSize{get; set; }
```

Property Type

System.Int32

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

RowsToFetchPerRoundTrip

This property specifies the total number of rows to retrieve per database round trip.

Declaration

```
// C#  
public static Int64 RowsToFetchPerRoundTrip { get; set; }
```

Property Type

System.Int64

Remarks

This property has no default value. If it is not set, then ODP.NET ignores this property.

The row data fetched in a single round trip applies to scalar types only, such as NUMBER and VARCHAR2 columns. If reference data types, such as LOBs, UDTs, and XMLType, exist in the result set, they are retrieved in separate round trips. These reference types can have their own fetch tuning parameters as reference type data sizes can vary in size from row to row, sometimes significantly.

RowsToFetchPerRoundTrip can be set before or after the OracleCommand executes on OracleConnection, OracleCommand, OracleDataReader, or OracleRefCursor. The value can also be changed after initial data fetches so that more or fewer rows are fetched on subsequent round trips. For OracleConfiguration, this property can only be set before the first connection opens.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

SelfTuning

This property specifies whether self-tuning is enabled for an ODP.NET application. By default, self-tuning (true) is enabled.

Declaration

```
// C#  
public static bool SelfTuning { get; set; }
```

Property Type

System.Boolean

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

SendBufferSize

This property specifies the buffer space limit for send operations of sessions.

Declaration

```
// C#  
public static int SendBufferSize { get; set; }
```

Property Type

System.Int32

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

StatementCacheSize

This property specifies the number of cursors or statements to be cached for each database connection. This setting corresponds to the connection string Statement Cache Size attribute. A value greater than zero enables statement caching. Default value is zero.

Declaration

```
// C#  
public static int StatementCacheSize { get; set; }
```

Property Type

System.Int32

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

TcpNoDelay

This property preempts delays in buffer flushing within the TCP/IP protocol stack. Default value is true.

Declaration

```
// C#  
public static bool TcpNoDelay { get; set; }
```

Property Type

System.Boolean

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

OracleConfiguration Diagnostics and Tracing Properties

OracleConfiguration Diagnostics and Tracing properties are listed in [Table 6-43](#).

Table 6-43 OracleConfiguration Diagnostics and Tracing Properties

Property	Description
ConnectionIdPrefix	Identifies a particular application's connections by appending its string to the connection identifier <i>Not Available in ODP.NET, Unmanaged Driver</i>
OpenTelemetryTracing	Specifies whether to dynamically enable or disable ODP.NET OpenTelemetry tracing instrumentation or not. <i>Not Available in ODP.NET, Unmanaged Driver</i>
PoolNames	Specifies a string to identify the monitored connection pool more easily
Program	Specifies the application program name <i>Not available in ODP.NET, Unmanaged Driver</i>
SuppressErrorURL	Shows or suppresses the Oracle Database Error Messages URL for the specific error with more information on its cause and resolution
TraceFileLocation	Specifies the destination directory to output provider traces
TraceFileMaxSize	Specifies the maximum file size of each trace file
TraceLevel	Specifies the generated trace level to trace ODP.NET calls and diagnose provider issues
TraceOption	Specifies whether to generate a single trace file or multiple trace files for multithreaded applications

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

ConnectionStringPrefix

This property is used to identify a particular application's connections by appending its string to the connection identifier. The identifier and prefix can be used to uniquely identify the application connections for tracing and diagnosis purposes.

Declaration

```
// C#  
public static string ConnectionIdPrefix { set; }
```

Property Value

System.String

Remarks

The property value is an 8-byte identifier limited to alphanumeric characters and underscore (_).

📘 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

OpenTelemetryTracing

This property specifies whether to dynamically enable or disable ODP.NET OpenTelemetry tracing instrumentation or not.

Declaration

```
// C#  
public static bool OpenTelemetryTracing { get; set; }
```

Property Type

Bool

Remarks

Default value is true.

This property can be changed at runtime to dynamically turn off and on ODP.NET OpenTelemetry tracing. When this OracleConfiguration property is explicitly set to a value, it overrides the OpenTelemetryTracing value set in the .NET configuration file.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

PoolNames

This property specifies a string to identify the monitored connection pool more easily.

Declaration

```
// C#  
public static OraclePoolNameCollection PoolNames { get; }
```

Property Type

OraclePoolNameCollection class instance

Remarks

OraclePoolNameCollection class exposes Add and Remove methods to add or remove pool name for each unique connection string. Example:

```
OracleConfiguration.PoolNames.Add("PoolA", "<unique connection string>");
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

Program

This property specifies the application program name.

Declaration

```
// C#  
public static string Program {get; set;}
```

Property Value

This property stores the name identifying the client program.

Remarks

This name is also stored in the PROGRAM column of V\$SESSION on the database server. By default, the application/process name obtained from .NET is used. The app can assign different value or reset to the default by assigning a null value.

The string is returned regardless of the connection state. This property cannot be changed after the first successfully opened connection.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)
- [OracleDataSourceCollection Class](#)

SuppressErrorURL

This property shows or suppresses the Oracle Database Error Messages URL for the specific error with more information on its cause and resolution.

Declaration

```
// C#  
public static bool SuppressErrorURL { get; set; }
```

Property Type

System.Boolean

Remarks

The default value is false.

When set to true, ODP.NET exception messages will only contain an error code, for example, ORA-12345, and a brief description. This is the same behavior as earlier ODP.NET versions.

When set to false, ODP.NET exception messages will include the error code, description, and an URL to the Oracle Database Error Messages website for the particular error code. This specific error code web page includes more information, such as the possible error cause and actions to resolve it.

The error URLs are not appended for ODP specific errors (ORA-50000 - ORA 50999)

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

TraceFileLocation

This property specifies the destination directory to output provider traces.

Declaration

```
// C#  
public static string TraceFileLocation { get; set; }
```

Property Type

System.String

Remarks

On Windows, the default TraceFileLocation is <Windows user temporary folder>\ODP.NET\core\trace. On Linux, the default is <current user temporary folder>/ODP.NET/core/trace. If you modify the trace file location, then do not use a write-protected directory, such as C:\ on Windows. Write-protection may prevent ODP.NET from being able to write a trace file to that particular directory location.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

TraceFileMaxSize

This property specifies the maximum file size of each trace file.

Declaration

```
// C#  
public static int TraceFileMaxSize { get; set; }
```

Property Type

System.Int32

Remarks

Administrators can maintain reasonably sized trace files by using this property to enforce a maximum size that any individual ODP.NET trace file can be. The TraceFileMaxSize value is in megabytes. If no value is set, TraceFileMaxSize defaults to a value of 100, which is 100 megabytes (MB).

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

TraceLevel

This property specifies the generated trace level to trace ODP.NET calls and diagnose provider issues. Errors will always be traced. Default value is 0 indicating tracing is disabled.

Declaration

```
// C#  
public static int TraceLevel { get; set; }
```

Property Type

System.Int32

Valid Values:

- 1 = public APIs
- 2 = private APIs
- 4 = network APIs/data
- 8 = disables writing SQL statements and network packet contents

These values can be bitwise ORed. To enable all traces, set TraceLevel to 7. Use TraceLevel 8 if you require SQL statements and network packet contents to be excluded from the trace.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

TraceOption

This property specifies whether to generate a single trace file or multiple trace files for multithreaded applications. Default value is 0 indicating single trace file for all application threads.

Declaration

```
// C#  
public static int TraceOption { get; set; }
```

Property Type

System.Int32

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConfiguration Class](#)
- [OracleConfiguration Members](#)

OracleConfigurationProvider Class

The OracleConfigurationProvider class contains enumerated values that specify the ODP.NET centralized configuration providers that the application is allowed to use.

Class Inheritance

System.Object

Oracle.ManagedDataAccess.OracleConfigurationProvider

Declaration

```
// C#  
public sealed class OracleConfigurationProvider : IComparable
```

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

Remarks

OracleConfigurationProvider is technically a class but is functionally an enumeration that has a predefined list of possible values. You use the bitwise OR operator '|' to combine multiple enum values.

This class is only applicable to the OracleConfiguration.OracleConfigurationProviders property; It cannot be in sqlnet.ora nor .NET configuration files.

Member Name	Description
None	Disables all configuration providers
Azure	Enables Azure App configuration provider
OCIObject	Enables Oracle Cloud Infrastructure (OCI) Configuration provider

Member Name	Description
File	Enable local file provider
AzureVault	Enables Azure Vault Configuration provider
OCIVault	Enables OCI Vault Configuration provider
All	Enables all configuration providers. This is the default value.

The property default value of All will remain constant in future ODP.NET versions. The configuration providers list supported by All may increase in the future.

Thread Safety

This class only has public static methods, so it is entirely thread safe.

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"

OracleConnection Class

An OracleConnection object represents a connection to an Oracle database.

① Note

The OracleConnection class has been extended and modified in the Oracle.ManagedDataAccess.Azure and Oracle.ManagedDataAccess.Oci assemblies. They expose methods for ODP.NET to connect through Azure and Oracle Cloud Infrastructure token authentication, respectively.

Class Inheritance

System.Object

System.MarshalByRefObject

System.ComponentModel.Component

System.Data.Common.DbConnection

Oracle.DataAccess.Client.OracleConnection

Declaration

```
// C#  
public sealed class OracleConnection : DbConnection, IDbConnection, ICloneable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;

class OracleConnectionSample
{
    static void Main()
    {
        // Connect
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        // Execute a SQL SELECT
        OracleCommand cmd = con.CreateCommand();
        cmd.CommandText = "select * from emp";
        OracleDataReader reader = cmd.ExecuteReader();

        // Print all employee numbers
        while (reader.Read())
            Console.WriteLine(reader.GetInt32(0));

        // Clean up
        reader.Dispose();
        cmd.Dispose();
        con.Dispose();
    }
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Members](#)
- [OracleConnection Constructors](#)
- [OracleConnection Static Methods](#)
- [OracleConnection Properties](#)
- [OracleConnection Public Methods](#)
- [OracleConnection Events](#)

OracleConnection Members

OracleConnection members are listed in the following tables.

OracleConnection Constructors

OracleConnection constructors are listed in [Table 6-44](#).

Table 6-44 OracleConnection Constructors

Constructor	Description
OracleConnection Constructors	Instantiates a new instance of the OracleConnection class (Overloaded)

OracleConnection Static Methods

The OracleConnection static methods are listed in [Table 6-45](#).

Table 6-45 OracleConnection Static Methods

Method	Description
ClearAllPools	Clears all connections from all the connection pools
ClearPool	Clears the connection pool that is associated with the provided OracleConnection object
ClearInteractiveTokenCache	Clears the connection pool, Microsoft Entra ID, and OCI IAM access tokens from the ODP.NET cache. <i>Not available in ODP.NET, Unmanaged Driver</i>
Equals	Inherited from System.Object (Overloaded)
UnregisterCloudConfigNotification	Unregisters an OracleConnection from receiving cloud provider change notifications <i>Not available in ODP.NET, Unmanaged Driver</i>

OracleConnection Properties

OracleConnection properties are listed in [Table 6-46](#).

Table 6-46 OracleConnection Properties

Property	Description
AccessToken	Used to get or set <code>OracleAccessToken</code> on the connection
ActionName	Specifies the action name for the connection
AllowCertificateSelectionUI	Indicates whether to allow the user to select a specific TLS/SSL certificate via a graphical interface for their database connection when using a MCS wallet location <i>Not Available in ODP.NET, Unmanaged Driver</i>
AutoCommit	Specifies whether to commit each SQL statement implicitly or only until an explicit commit occurs
AutoProxy	Enables using a HTTPS proxy server for the ODP.NET connection
BindByName	Specifies the binding method in the collection <i>Not Available in ODP.NET, Unmanaged Driver</i>
ChunkMigrationConnectionTimeout	This setting pauses the connection request until the migration completes or fails to migrate in the specified time, then connect to the correct shard with the data requested. <i>Not Available in ODP.NET, Unmanaged Driver</i>
ClientId	Specifies the client identifier for the connection
ClientInfo	Specifies the client information for the connection
ClientTimeZoneFileVersion	Returns the highest time zone file version in the <code>TimeZoneFileLocation</code> directory and embedded in ODP.NET
CommandTimeout	Specifies the number of seconds the command is allowed to execute before terminating the execution with an exception <i>Not Available in ODP.NET, Unmanaged Driver</i>
ConnectionIdPrefix	Identifies a particular application's connections by appending its string to the connection identifier <i>Not Available in ODP.NET, Unmanaged Driver</i>
ConnectionString	Specifies connection information or URL used to connect to an Oracle database
ConnectionTimeout	Indicates the maximum amount of time that the <code>Open</code> method can take to obtain a pooled connection before the request is terminated
ConnectionType	Determines whether a particular connection object is associated with a TimesTen database connection, an Oracle database connection, or no physical connection <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
Container	Inherited from <code>System.ComponentModel.Component</code>
Credential	Used to get or set an <code>OracleCredential</code> object on the connection

Table 6-46 (Cont.) OracleConnection Properties

Property	Description
Database	<i>Not Supported</i>
DatabaseCharset	Specifies the CHAR database character set on the connection and is used to optimize connection establishment <i>Not Available in ODP.NET, Unmanaged Driver</i>
DatabaseNCharset	Specifies the NCHAR database character set on the connection and is used to optimize connection establishment <i>Not Available in ODP.NET, Unmanaged Driver</i>
DatabaseDomainName	Specifies the name of the database domain to which the connection is set
DatabaseEditionName	Sets or gets the edition name for the OracleConnection object
DatabaseMaxIdentifierLength	Specifies the maximum identifier length the connection supports
DatabaseName	Specifies the name of the database to which the connection is set
DatabaseTimeZoneFileVersion	Returns the database's time zone file version
DataSource	Specifies the Oracle Net Services Name, Connect Descriptor, or an easy connect naming that identifies the database to which to connect
DRCPConnectionClass	Specifies a logical name that identifies the DRCP connection pool that the ODP.NET connection will use
DRCPPurity	Specifies whether an application can reuse a pooled session (Pooled) or must use a new session (New)
FetchSize	Specifies the total memory size, in bytes, that the provider allocates to cache data fetched in one database round-trip <i>Not Available in ODP.NET, Unmanaged Driver</i>
HostName	Specifies the name of the host to which the connection is set
HttpsProxy	Specifies the HTTPS proxy server
HttpsProxyPort	Specifies the HTTPS proxy server port number
InstanceName	Specifies the name of the instance to which the connection is set
IsInTransaction	Specifies if the current application is in a transaction
IsNNEChecksummed	Indicates if the connection uses Oracle Database Native Network Encryption (NNE) checksumming <i>Not Available in ODP.NET, Unmanaged Driver</i>
IsNNEEncrypted	Indicates if the connection uses Oracle Database Native Network Encryption (NNE). <i>Not Available in ODP.NET, Unmanaged Driver</i>

Table 6-46 (Cont.) OracleConnection Properties

Property	Description
IsTLSEncrypted	Indicates if the connection uses TCPS, SSL, or TLS <i>Not Available in ODP.NET, Unmanaged Driver</i>
KeepAlive	Specifies whether to keep an idle connection alive. <i>Not Available in ODP.NET, Unmanaged Driver</i>
KeepAliveInterval	Specifies the time interval after an unacknowledged probe before a new probe is sent. <i>Not Available in ODP.NET, Unmanaged Driver</i>
KeepAliveTime	Specifies the idle time before a keepalive probe is sent. <i>Not Available in ODP.NET, Unmanaged Driver</i>
MaxStatementCacheSize	Specifies the maximum number of statements that can be cached when self-tuning is enabled <i>Not Available in ODP.NET, Unmanaged Driver</i>
ModuleName	Specifies the module name for the connection
NamesDirectoryPath	Specifies the naming method order used for client name resolution lookups
OciCompartment	Gets or sets the Oracle Cloud Identifier (OCID) of the target database's compartment
OciDatabase	Gets or sets the target database's Oracle Cloud Identifier (OCID)
OciIamUrl	Gets or sets the URL end point for getting the database token
OciTenacy	Gets or sets the Oracle Cloud Identifier (OCID) of the IAM user's cloud tenancy to retrieve the database token
PasswordAuthentication	Sets the connection PASSWORD_AUTH parameter value
PDBName	Used to set and get the name of the pluggable database to which the session is connected to
Pipelining	Enables asynchronous execution on the database server side <i>Not Available in ODP.NET, Unmanaged Driver</i>
PoolName	Specifies the name that uniquely identifies the pool in .NET metrics or performance counters. <i>Not available in ODP.NET, Unmanaged Driver and ODP.NET, Managed Driver</i>
ProviderName	Specifies the ODP.NET provider type and version <i>Not Available in ODP.NET, Unmanaged Driver</i>
ProviderNuGetVersion	Specifies the ODP.NET NuGet version number <i>Not Available in ODP.NET, Unmanaged Driver</i>
ProviderVersion	Specifies the ODP.NET assembly version
RemoteConfigurationFiltering	Specifies whether local configuration file access, such as to wallet and tnsnames.ora files, is allowed from centralized configuration providers

Table 6-46 (Cont.) OracleConnection Properties

Property	Description
Roles	Retrieves the <code>OracleRoleCollection</code> object associated with the connection. <i>Not Available in ODP.NET, Unmanaged Driver</i>
RowsToFetchPerRoundTrip	Specifies the total number of rows to retrieve per database round trip <i>Not Available in ODP.NET, Unmanaged Driver</i>
SDU	Specifies the session data unit size (SDU) for Oracle networking communication between client and server
ServerVersion	Specifies the version number of the Oracle database to which the <code>OracleConnection</code> has established a connection
ServiceName	Specifies the name of the service to which the connection is set
ServiceRelocationConnectionTimeout	Specifies the time to wait before retrying connecting to a service that becomes unavailable <i>Not Available in ODP.NET, Unmanaged Driver</i>
SessionlessTransactionOptions	Indicates the sessionless transaction options <i>Not Available in ODP.NET, Unmanaged Driver</i>
SessionlessTransactionId	Returns the sessionless transaction identifier <i>Not Available in ODP.NET, Unmanaged Driver</i>
Site	Inherited from <code>System.ComponentModel.Component</code>
SqlNetAllowedLogonVersionClient	Specifies the minimum authentication protocol that is to be used for a given <code>OracleConnection</code> object
SqlNetAuthenticationServices	This property enables one or more authentication services, such as TCP/IP with SSL.
SqlNetCryptoChecksumClient	Specifies the checksum client behavior. <i>Not Available in ODP.NET, Unmanaged Driver</i>
SqlNetCryptoChecksumTypesClient	Specifies the checksum algorithms the client can use. <i>Not Available in ODP.NET, Unmanaged Driver</i>
SqlNetEncryptionClient	Specifies the encryption client behavior <i>Not Available in ODP.NET, Unmanaged Driver</i>
SqlNetEncryptionTypesClient	Specifies encryption algorithms that the client can use. <i>Not Available in ODP.NET, Unmanaged Driver</i>
SqlNetWalletOverride	Specifies whether the client overrides the strong authentication credential with the stored wallet password credential for the database connection
SSLCertificateThumbprint	Specifies the TLS/SSL certificate thumbprint in the certificate store for the ODP.NET connection to use
SSLServerCertDN	Sets the distinguished name (DN) string that will be used to validate the server certificate DN

Table 6-46 (Cont.) OracleConnection Properties

Property	Description
SSLServerDNMatch	Enables server certificate distinguished name (DN) validation
SSLVersion	Enforces the connection to use a specific SSL/TLS version <i>Not available in ODP.NET, Unmanaged Driver</i>
State	Specifies the current state of the connection
StatementCacheSize	Specifies the current size of the statement cache associated with this connection
SuppressGetDecimalInvalidCastException	Specifies whether to suppress the <code>InvalidCastException</code> and return a rounded-off 28 or 29 precision Oracle NUMBER value that can be represented as a .NET Decimal <i>Not available in ODP.NET, Unmanaged Driver</i>
SwitchedConnection	If the returned pooled connection changed its service name (<code>ServiceName</code>) or pluggable database (<code>PDBName</code>) in order to connect to the desired pluggable database, then this property will consider the connection switched
TAFMode	Configures Transparent Application Failover behavior on the connection <i>Not available in ODP.NET, Unmanaged Driver</i>
TLSExcludeVersion	Indicates which SSL/TLS versions connections should not be enabled for. <i>Not Available in ODP.NET, Unmanaged Driver</i>
TLSVersion	Enforces the connection to use a specific SSL/TLS version <i>Not Available in ODP.NET, Unmanaged Driver</i>
TnsAdmin	Specifies the directory location of <code>tnsnames.ora</code> or <code>sqlnet.ora</code> , or both <i>Not Available in ODP.NET, Unmanaged Driver</i>
TokenAuthentication	Sets the value for <code>TOKEN_AUTH</code> parameter for the connection <i>Not Available in ODP.NET, Unmanaged Driver</i>
TokenLocation	Gets or sets the token location path for the connection where the signature token and private key files are located <i>Not Available in ODP.NET, Unmanaged Driver</i>
UseClientInitiatedCQN	Specifies whether to use Client Initiated Continuous Query Notification (CICQN) or traditional Continuous Query Notification (CQN) that does not rely on a persistent connection <i>Not Available in ODP.NET, Unmanaged Driver</i>

Table 6-46 (Cont.) OracleConnection Properties

Property	Description
UseHourOffsetForUnsupportedTimezone	Specifies whether the hour offset can be used for the session time zone, when the Oracle time zone region name that is associated with the .NET locale is not supported by the Oracle database being used <i>Not Available in ODP.NET, Unmanaged Driver</i>
UseSNI	Specifies whether to use Server Name Indication (SNI) <i>Not Available in ODP.NET, Unmanaged Driver</i>
WalletLocation	Specifies the location of wallets. Wallets are certificates, keys, and trustpoints processed by SSL/TLS <i>Not Available in ODP.NET, Unmanaged Driver</i>
WalletPassword	Stores the password to decrypt the PKCS #12 wallet <i>Not Available in ODP.NET, Unmanaged Driver</i>

OracleConnection Public Methods

OracleConnection public methods are listed in [Table 6-47](#).

Table 6-47 OracleConnection Public Methods

Public Method	Description
BeginRequest	Starts an explicit request boundary when using Application Continuity <i>Not available in ODP.NET, Unmanaged Driver</i>
BeginSessionlessTransaction	Starts a new sessionless transaction and returns its global transaction identifier <i>Not available in ODP.NET, Unmanaged Driver</i>
BeginTransaction	Begins a local transaction (Overloaded)
ChangeDatabase	<i>Not Supported</i>
Clone	Creates a copy of an OracleConnection object
Close	Closes the database connection
CreateCommand	Creates and returns an OracleCommand object associated with the OracleConnection object
CreateObjRef	Inherited from System.MarshalByRefObject
Commit()	Commits the database transaction
DisableReplay	Disables replay on the connection in Application Continuity <i>Not available in ODP.NET, Unmanaged Driver</i>
Dispose	Inherited from System.ComponentModel.Component
EndRequest	Indicates an ending explicit request boundary when using Application Continuity <i>Not available in ODP.NET, Unmanaged Driver</i>

Table 6-47 (Cont.) OracleConnection Public Methods

Public Method	Description
EnlistDistributedTransaction	Enables applications to explicitly enlist in a specified distributed transaction <i>Not supported in .NET Core</i>
EnlistTransaction	Enables applications to enlist in a specified distributed transaction
Equals	Inherited from System.Object (Overloaded)
FlushCache	Flushes all updates and deletes made through REF objects retrieved using this connection <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
GetHashCode	Inherited from System.Object
GetLifetimeService	Inherited from System.MarshalByRefObject
GetSchema	Returns schema information for the data source of the OracleConnection
GetSessionInfo	Returns or refreshes the property values of the OracleGlobalization object that represents the globalization settings of the session (Overloaded)
GetType	Inherited from System.Object
InitializeLifetimeService	Inherited from System.MarshalByRefObject
Open	Opens a database connection with the property settings specified by the ConnectionString
OpenAsync	Returns a Task-based asynchronous version of OracleConnection.Open (Overloaded)
OpenWithNewClientSecret	Opens a new connection using a new Microsoft Entra ID client secret <i>Not available in ODP.NET, Unmanaged Driver</i>
OpenWithNewPassword	Opens a new connection with the new password (Overloaded)
OpenWithNewPasswordAsync	A Task-based asynchronous version of OracleConnection.OpenWithNewPassword(), which opens a new connection with the new password. (Overloaded)
OpenWithNewToken	This method can be used to open a new connection by providing the refreshed/updated access token directly to ODP.NET (Overloaded)
PurgeStatementCache	Flushes the Statement Cache by closing all open cursors on the database, when statement caching is enabled
ResumeSessionlessTransaction	Resumes a sessionless transaction using the provided unique transaction identifier <i>Not Available in ODP.NET, Unmanaged Driver</i>
Rollback	Rolls back the local transaction
Save(String)	Creates a savepoint in the database
SetSessionInfo	Alters the session's globalization settings with the property values provided by the OracleGlobalization object

Table 6-47 (Cont.) OracleConnection Public Methods

Public Method	Description
SetShardingKey(OracleShardingKey, OracleShardingKey)	Enables applications to set the sharding key and super sharding key before requesting a connection
SuspendSessionlessTransaction	Suspends the active sessionless transaction immediately from the session. <i>Not available in ODP.NET, Unmanaged Driver</i>
ToString	Inherited from System.Object
UseAzureTokenAuthentication	Associates the Microsoft Entra ID token authentication settings with the OracleConnection object for use upon the next connection open <i>Not available in ODP.NET, Unmanaged Driver</i>
UseOciTokenAuthentication	Associates the OCI IAM token authentication settings with the OracleConnection object for use upon the next connection open <i>Not available in ODP.NET, Unmanaged Driver</i>

OracleConnection Events

OracleConnection events are listed in [Table 6-48](#).

Table 6-48 OracleConnection Events

Event Name	Description
ConnectionOpen	This event is triggered upon the OracleConnection.Open() method
Disposed	Inherited from System.ComponentModel.Component
Failover	An event that is triggered when an Oracle failover occurs <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
HAEvent	An event that is triggered when an HA event occurs
InfoMessage	An event that is triggered for any message or warning sent by the database
StateChange	An event that is triggered when the connection state changes

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)

OracleConnection Constructors

OracleConnection constructors instantiate new instances of the OracleConnection class.

Overload List:

- [OracleConnection\(\)](#)
This constructor instantiates a new instance of the OracleConnection class using default property values.
- [OracleConnection\(string\)](#)
This constructor instantiates a new instance of the OracleConnection class with the provided connection string.
- [OracleConnection\(string, OracleCredential\)](#)
This constructor instantiates a new OracleConnection class instance using the provided connection string and OracleCredential class.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

OracleConnection()

This constructor instantiates a new instance of the OracleConnection class using default property values.

Declaration

```
// C#  
public OracleConnection();
```

Remarks

The properties for OracleConnection are set to the following default values:

- ConnectionString = empty string
- ConnectionTimeout = 15 (default value of 0 is used for the implicit database connection)
- DataSource = empty string
- ServerVersion = empty string

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

OracleConnection(string)

This constructor instantiates a new instance of the `OracleConnection` class with the provided connection string.

Declaration

```
// C#  
public OracleConnection(string connectionString);
```

Parameters

- *connectionString*
The connection information used to connect to the Oracle database.

Remarks

The `ConnectionString` property is set to the supplied *connectionString*. The `ConnectionString` property is parsed and an exception is thrown if it contains invalid connection string attributes or attribute values.

The properties of the `OracleConnection` object default to the following values unless they are set by the connection string:

- `ConnectionString` = empty string
- `ConnectionTimeout` = 15 (default value of 0 is used for the implicit database connection)
- `DataSource` = empty string
- `ServerVersion` = empty string

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

OracleConnection(string, OracleCredential)

This constructor instantiates a new `OracleConnection` class instance using the provided connection string and `OracleCredential` class.

Declaration

```
// C#  
public OracleConnection(string connectionString, OracleCredential orclCredential);
```

Parameters

- *connectionString*
connection string that does not contain any of user `id`, password, DBA Privilege, proxy user `id`, nor proxy password.

- `orclCredential`

OracleCredential object containing user credentials. If this parameter is passed as null then the behavior of OracleConnection will be same as OracleConnection with normal connection string.

Remarks

Use this constructor to create a new OracleConnection object with an OracleCredential object containing user credentials and a connection string that does not contain any of user id, password, DBA Privilege, proxy user id, nor proxy password.

ODP.NET uses a modified connection pooling algorithm with OracleCredential, which is described in more detail in the section [Connection Pooling with OracleCredential](#).

Exceptions

InvalidOperationException is raised when non-null OracleCredential object is used with a connection string containing any of user id, password, DBA Privilege, proxy user id, or proxy password.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

OracleConnection Static Methods

The OracleConnection static methods are listed in [Table 6-49](#).

Table 6-49 OracleConnection Static Methods

Method	Description
ClearAllPools	Clears all connections from all the connection pools
ClearPool	Clears the connection pool that is associated with the provided OracleConnection object.
ClearInteractiveTokenCache	Clears the connection pool, Microsoft Entra ID, and OCI IAM access tokens from the ODP.NET cache. <i>Not available in ODP.NET, Unmanaged Driver</i>
Equals	Inherited from System.Object (Overloaded)
UnregisterCloudConfigNotification	Unregisters an OracleConnection from receiving cloud provider change notifications <i>Not available in ODP.NET, Unmanaged Driver</i>

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

ClearAllPools

This method clears all connections from all the connection pools.

Declaration

```
// C#  
public static void ClearAllPools();
```

Remarks

This call is analogous to calling `ClearPool` for all the connection pools that are created for the application.

For connection string URLs, this method removes all existing URLs from the ODP.NET cache. Upon the next connection attempt, ODP.NET will refresh the connection information from the centralized configuration provider.

Exceptions

`InvalidOperationException` – No connection pool could be found for the application.

Example

```
// C#  
// Sample demonstrating the use of ClearAllPools API in OracleConnection class  
  
using System;  
using Oracle.DataAccess.Client;  
  
class ClearAllPoolsSample  
{  
    static void Main()  
    {  
        Console.WriteLine("Running ClearAllPools sample..." );  
        // Set the connection string  
        string strConn = "User Id=scott;Password=tiger;Data Source=oracle;" +  
            "Min pool size=5;";  
        OracleConnection conn = new OracleConnection(strConn);  
  
        // Create another connection object with a different connection string  
        string strConnNew = "User Id=scott;Password=tiger;Data Source=oracle;";  
        OracleConnection connNew = new OracleConnection(strConnNew);  
  
        // Open the connections. Separate pools are created for conn and connNew  
        conn.Open();  
        connNew.Open();  
  
        // Clears the pools associated with conn and connNew  
        OracleConnection.ClearAllPools ();  
    }  
}
```

```
// cleanup
conn.Close();
connNew.Close();
Console.WriteLine("Done!");
}
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- ["ClearPool"](#)

ClearPool

This method clears the connection pool that is associated with the provided `OracleConnection` object.

Declaration

```
// C#
public static void ClearPool(OracleConnection connection);
```

Remarks

When this method is invoked, all idle connections are closed and freed from the pool. Currently used connections are not discarded until they are returned to the pool.

`ClearPool` does not automatically repopulate the pool with new connections. This prevents the pool from being repopulated with invalid connections if client remains unable to connect with the database server. Developers programmatically control when the pool is repopulated by calling `OracleConnection.Open()`, which will repopulate the pool with at least the Min Pool Size number of connections.

Connections created after this method invocation are not cleared unless another invocation is made.

Starting with ODP.NET 21c, `ClearPool` will now clear the pool even if ODP.NET is in the process of populating the pool to the Min Pool Size number of connections.

This method can be invoked with an `OracleConnection` object before opening the connection as well as after, provided the `ConnectionString` is properly set.

For connection string URLs, this method removes existing URLs from the ODP.NET cache. Upon the next connection attempt, ODP.NET will refresh the connection information from the centralized configuration provider.

Exceptions

`InvalidOperationException` – Either the connection pool cannot be found or the provided connection string is invalid.

Example

```
// C#
// Sample demonstrating the use of ClearPool API in OracleConnection class

using System;
using Oracle.DataAccess.Client;

class ClearPoolSample
{
    static void Main()
    {
        Console.WriteLine("Running ClearPool sample..." );
        // Set the connection string
        string strConn = "User Id=scott;Password=tiger;Data Source=oracle;" +
                        "Min pool size=5;";
        OracleConnection conn = new OracleConnection(strConn);

        // Open the connection
        conn.Open();

        // Clears the connection pool associated with connection 'conn'
        OracleConnection.ClearPool (conn);

        // This connection will be placed back into the pool
        conn.Close ();

        // Open the connection again to create additional connections in the pool
        conn.Open();

        // Create a new connection object
        OracleConnection connNew = new OracleConnection(strConn);

        // Clears the pool associated with Connection 'connNew'
        // Since the same connection string is set for both the connections,
        // connNew and conn, they will be part of the same connection pool.
        // We need not do an Open() on the connection object before calling
        // ClearPool
        OracleConnection.ClearPool (connNew);

        // cleanup
        conn.Close();
        Console.WriteLine("Done!");
    }
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

ClearInteractiveTokenCache

The static method clears the connection pool, Microsoft Entra ID, and OCI IAM access tokens from the ODP.NET cache.

Declaration

```
// C#  
public void static ClearInteractiveTokenCache();
```

Description

Invoking this method removes all the Microsoft Entra ID and OCI IAM access tokens from ODP.NET's cache so that they cannot be re-used when creating a new connection pool at the application level. Users will be prompted for login credentials upon the next connection request.

This method also clears the access token cache on the connection pool level, which means existing pools can not use the cached tokens.

Existing open connections do not use cached tokens and will continue to work. Only new connection requests have to acquire new cached tokens to connect.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

UnregisterCloudConfigNotification

This method unregisters an OracleConnection from receiving cloud provider change notifications.

Declaration

```
// C#  
public static void UnregisterCloudConfigNotification(OracleConnection con);
```

Remarks

This method unregisters connections that use connection URLs from receiving cloud provider change notifications. From a pooled connection, this method unregisters all connections within that pool. This method can be called from open or closed connections.

Exceptions

None.

Code Sample

```
// ODP.NET for Azure App Configuration code sample  
string connStr = "config-azure://myAppConfig?key=test/  
&SERVICE_BUS_HOST={serviceBusHost}&SERVICE_BUS_QUEUE={queueName}";  
OracleConnection conn = new OracleConnection(connStr);  
// Connection is opened and registered for change notifications  
conn.Open();  
  
// Unregister from change notifications  
OracleConnection.UnregisterCloudConfigNotifications(conn);
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

OracleConnection Properties

OracleConnection properties are listed in [Table 6-50](#)

Table 6-50 OracleConnection Properties

Property	Description
AccessToken	Used to get or set OracleAccessToken on the connection
ActionName	Specifies the action name for the connection
AllowCertificateSelectionUI	Indicates whether to allow the user to select a specific TLS/SSL certificate via a graphical interface for their database connection when using a MCS wallet location <i>Not Available in ODP.NET, Unmanaged Driver</i>
AutoCommit	Specifies whether to commit each SQL statement implicitly or only until an explicit commit occurs
AutoProxy	Enables using a HTTPS proxy server for the ODP.NET connection
BindByName	Specifies the binding method in the collection <i>Not Available in ODP.NET, Unmanaged Driver</i>
ChunkMigrationConnectionTimeout	This setting pauses the connection request until the migration completes or fails to migrate in the specified time, then connect to the correct shard with the data requested. <i>Not Available in ODP.NET, Unmanaged Driver</i>
ClientId	Specifies the client identifier for the connection
ClientInfo	Specifies the client information for the connection
ClientTimeZoneFileVersion	Returns the highest time zone file version in the TimeZoneFileLocation directory and embedded in ODP.NET
CommandTimeout	Specifies the number of seconds the command is allowed to execute before terminating the execution with an exception <i>Not Available in ODP.NET, Unmanaged Driver</i>
ConnectionIdPrefix	Identifies a particular application's connections by appending its string to the connection identifier <i>Not Available in ODP.NET, Unmanaged Driver</i>
ConnectionString	Specifies connection information or URL used to connect to an Oracle database

Table 6-50 (Cont.) OracleConnection Properties

Property	Description
ConnectionTimeout	Indicates the maximum amount of time that the Open method can take to obtain a pooled connection before the request is terminated
ConnectionType	Determines whether a particular connection object is associated with a TimesTen database connection, an Oracle database connection, or no physical connection <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
Container	Inherited from System.ComponentModel.Component
Credential	Used to get or set an OracleCredential object on the connection
Database	<i>Not Supported</i>
DatabaseCharset	Specifies the CHAR database character set on the connection and is used to optimize connection establishment <i>Not Available in ODP.NET, Unmanaged Driver</i>
DatabaseNCharset	Specifies the NCHAR database character set on the connection and is used to optimize connection establishment <i>Not Available in ODP.NET, Unmanaged Driver</i>
DatabaseDomainName	Specifies the name of the database domain to which the connection is set
DatabaseEditionName	Sets or gets the edition name for the OracleConnection object
DatabaseMaxIdentifierLength	Specifies the maximum identifier length the connection supports
DatabaseName	Specifies the name of the database to which the connection is set
DatabaseTimeZoneFileVersion	Returns the database's time zone file version
DataSource	Specifies the Oracle Net Services Name, Connect Descriptor, or an easy connect naming that identifies the database to which to connect
DRCPConnectionClass	Specifies a logical name that identifies the DRCP connection pool that the ODP.NET connection will use
DRCPPurity	Specifies whether an application can reuse a pooled session (Pooled) or must use a new session (New)
FetchSize	Specifies the total memory size, in bytes, that the provider allocates to cache data fetched in one database round-trip <i>Not Available in ODP.NET, Unmanaged Driver</i>
HostName	Specifies the name of the host to which the connection is set

Table 6-50 (Cont.) OracleConnection Properties

Property	Description
HttpsProxy	Specifies the HTTPS proxy server
HttpsProxyPort	Specifies the HTTPS proxy server port number
InstanceName	Specifies the name of the instance to which the connection is set
IsInTransaction	Specifies if the current application is in a transaction
IsNNEChecksummed	Indicates if the connection uses Oracle Database Native Network Encryption (NNE) checksumming <i>Not Available in ODP.NET, Unmanaged Driver</i>
IsNNEEncrypted	Indicates if the connection uses Oracle Database Native Network Encryption (NNE). <i>Not Available in ODP.NET, Unmanaged Driver</i>
IsTLSEncrypted	Indicates if the connection uses TCPS, SSL, or TLS <i>Not Available in ODP.NET, Unmanaged Driver</i>
KeepAlive	Specifies whether to keep an idle connection alive. <i>Not Available in ODP.NET, Unmanaged Driver</i>
KeepAliveInterval	Specifies the time interval after an unacknowledged probe before a new probe is sent. <i>Not Available in ODP.NET, Unmanaged Driver</i>
KeepAliveTime	Specifies the idle time before a keepalive probe is sent. <i>Not Available in ODP.NET, Unmanaged Driver</i>
MaxStatementCacheSize	Specifies the maximum number of statements that can be cached when self-tuning is enabled <i>Not Available in ODP.NET, Unmanaged Driver</i>
ModuleName	Specifies the module name for the connection
NamesDirectoryPath	Specifies the naming method order used for client name resolution lookups
OciCompartment	Gets or sets the Oracle Cloud Identifier (OCID) of the target database's compartment
OciDatabase	Gets or sets the target database's Oracle Cloud Identifier (OCID)
OciIamUrl	Gets or sets the URL end point for getting the database token
OciTenacy	Gets or sets the Oracle Cloud Identifier (OCID) of the IAM user's cloud tenancy to retrieve the database token
PasswordAuthentication	Sets the connection PASSWORD_AUTH parameter value
PDBName	Used to set and get the name of the pluggable database to which the session is connected to

Table 6-50 (Cont.) OracleConnection Properties

Property	Description
Pipelining	Enables asynchronous execution on the database server side <i>Not Available in ODP.NET, Unmanaged Driver</i>
PoolName	Specifies the name that uniquely identifies the pool in .NET metrics or performance counters. <i>Not available in ODP.NET, Unmanaged Driver and ODP.NET, Managed Driver</i>
ProviderName	Specifies the ODP.NET provider type and version <i>Not Available in ODP.NET, Unmanaged Driver</i>
ProviderNuGetVersion	Specifies the ODP.NET NuGet version number <i>Not Available in ODP.NET, Unmanaged Driver</i>
ProviderVersion	Specifies the ODP.NET assembly version
RemoteConfigurationFiltering	Specifies whether local configuration file access, such as to wallet and tnsnames.ora files, is allowed from centralized configuration providers
Roles	Retrieves the OracleRoleCollection object associated with the connection. <i>Not Available in ODP.NET, Unmanaged Driver</i>
RowsToFetchPerRoundTrip	Specifies the total number of rows to retrieve per database round trip <i>Not Available in ODP.NET, Unmanaged Driver</i>
SDU	Specifies the session data unit size (SDU) for Oracle networking communication between client and server
ServerVersion	Specifies the version number of the Oracle database to which the OracleConnection has established a connection
ServiceName	Specifies the name of the service to which the connection is set
ServiceRelocationConnectionTimeout	Specifies the time to wait before retrying connecting to a service that becomes unavailable <i>Not Available in ODP.NET, Unmanaged Driver</i>
SessionlessTransactionOptions	Indicates the sessionless transaction options <i>Not Available in ODP.NET, Unmanaged Driver</i>
SessionlessTransactionId	Returns the sessionless transaction identifier <i>Not Available in ODP.NET, Unmanaged Driver</i>
Site	Inherited from System.ComponentModel.Component
SqlNetAllowedLogonVersionClient	Specifies the minimum authentication protocol that is to be used for a given OracleConnection object
SqlNetAuthenticationServices	This property enables one or more authentication services, such as TCP/IP with SSL.
SqlNetEncryptionClient	Specifies the encryption client behavior <i>Not Available in ODP.NET, Unmanaged Driver</i>

Table 6-50 (Cont.) OracleConnection Properties

Property	Description
SqlNetEncryptionTypesClient	Specifies encryption algorithms that the client can use. <i>Not Available in ODP.NET, Unmanaged Driver</i>
SqlNetCryptoChecksumClient	Specifies the checksum client behavior. <i>Not Available in ODP.NET, Unmanaged Driver</i>
SqlNetCryptoChecksumTypesClient	Specifies the checksum algorithms the client can use. <i>Not Available in ODP.NET, Unmanaged Driver</i>
SqlNetWalletOverride	Specifies whether the client overrides the strong authentication credential with the stored wallet password credential for the database connection
SSLCertificateThumbprint	Specifies the TLS/SSL certificate thumbprint in the certificate store for the ODP.NET connection to use
SSLServerCertDN	Sets the distinguished name (DN) string that will be used to validate the server certificate DN
SSLServerDNMatch	Enables server certificate distinguished name (DN) validation
SSLVersion	Enforces the connection to use a specific SSL/TLS version <i>Not available in ODP.NET, Unmanaged Driver</i>
State	Specifies the current state of the connection
StatementCacheSize	Specifies the current size of the statement cache associated with this connection
SuppressGetDecimalInvalidCastException	Specifies whether to suppress the <code>InvalidCastException</code> and return a rounded-off 28 or 29 precision Oracle NUMBER value that can be represented as a .NET Decimal <i>Not available in ODP.NET, Unmanaged Driver</i>
SwitchedConnection	If the returned pooled connection changed its service name (<code>ServiceName</code>) or pluggable database (<code>PDBName</code>) in order to connect to the desired pluggable database, then this property will consider the connection switched.
TAFMode	Configures Transparent Application Failover behavior on the connection <i>Not available in ODP.NET, Unmanaged Driver</i>
TLSExcludeVersion	Indicates which SSL/TLS versions connections should not be enabled for. <i>Not Available in ODP.NET, Unmanaged Driver</i>
TLSVersion	Enforces the connection to use a specific SSL/TLS version <i>Not Available in ODP.NET, Unmanaged Driver</i>
TnsAdmin	Specifies the directory location of <code>tnsnames.ora</code> or <code>sqlnet.ora</code> , or both <i>Not Available in ODP.NET, Unmanaged Driver</i>

Table 6-50 (Cont.) OracleConnection Properties

Property	Description
TokenAuthentication	Sets the value for TOKEN_AUTH parameter for the connection <i>Not Available in ODP.NET, Unmanaged Driver</i>
TokenLocation	Gets or sets the token location path for the connection where the signature token and private key files are located <i>Not Available in ODP.NET, Unmanaged Driver</i>
UseClientInitiatedCQN	Specifies whether to use Client Initiated Continuous Query Notification (CICQN) or traditional Continuous Query Notification (CQN) that does not rely on a persistent connection <i>Not Available in ODP.NET, Unmanaged Driver</i>
UseHourOffsetForUnsupportedTimezone	Specifies whether the hour offset can be used for the session time zone, when the Oracle time zone region name that is associated with the .NET locale is not supported by the Oracle database being used <i>Not Available in ODP.NET, Unmanaged Driver</i>
UseSNI	Specifies whether to use Server Name Indication (SNI) <i>Not Available in ODP.NET, Unmanaged Driver</i>
WalletLocation	Specifies the location of wallets. Wallets are certificates, keys, and trustpoints processed by SSL/TLS <i>Not Available in ODP.NET, Unmanaged Driver</i>
WalletPassword	Stores the password to decrypt the PKCS #12 wallet <i>Not Available in ODP.NET, Unmanaged Driver</i>

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

AccessToken

This property is used to get or set OracleAccessToken on the connection.

Declaration

```
// C#
public OracleAccessToken AccessToken { get; set; }
```

Property Value

An `OracleAccessToken` object containing a database token and private key, or an access token, used for opening a connection.

Exceptions

- An `InvalidOperationException` will be raised if
 - the connection is already open,
 - the value of the `TokenAuthentication` property is not compatible with the type of token being set. This is applicable for managed ODP.NET and ODP.NET Core only,
 - `TokenAuthentication` has been set by the application as disabled, or the `TokenLocation` property is set. This is applicable for managed ODP.NET and ODP.NET Core only.
 - the `Credential` property is set,
 - the user or proxy information in the connection string is not compatible with token authentication,
 - user id value is other than /,
 - password, proxy user id, and proxy password is set.
- `ArgumentNullException` if any argument is null.

Description

If the token represents the same identity, then the same `OracleAccessToken` object should be set on the `OracleConnection`. The `OracleAccessToken` object, not the token, is used to identify the connection pool.

If an access token has expired, then `OracleConnection.OpenWithNewToken` must be called to open a new connection. Since the `OracleAccessToken` object cannot be modified, it is not possible to specify a refreshed access token on the `AccessToken` property. However, existing connections can still be dispensed, even after the access token has expired.

In unmanaged ODP.NET, if `CPVersion` is not set to any value, `AccessToken` property is set by the application, and `Open()` method is invoked, then the connection will set the value of `CPVersion` to 2.0. `CPVersion` cannot be set to 1.0 in this scenario. Doing so will result in an exception.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

ActionName

This property specifies the action name for the connection.

Declaration

```
// C#  
public string ActionName {set;}
```

Property Value

The string to be used as the action name.

Remarks

The default value is null.

Using the ActionName property allows the application to set the action name in the application context for a given OracleConnection object.

For unmanaged ODP.NET with CPVersion = 1.0, the ActionName property is reset to null upon the next server roundtrip following a Close or Dispose method call on the OracleConnection object. For all other ODP.NET provider types and CPVersion settings, ActionName property is reset to null when the Close or Dispose method is called on the OracleConnection object.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- ["Client Identifier and End-to-End Tracing"](#)
- *Oracle Database Security Guide*

AllowCertificateSelectionUI

This property indicates whether to allow the user to select a specific TLS/SSL certificate via a graphical interface for their database connection when using a MCS wallet location.

Declaration

```
// C#  
public Boolean AllowCertificateSelectionUI { get; set; }
```

Property Type

System.Boolean

Remarks

The property default value is false.

This property generates a pop up window for the end user to select a TLS/SSL certificate only when all of the following conditions are met:

- AllowCertificateSelectionUI is set to true
- App is running on Windows operating system

- App has specified a Microsoft Certificate Store (MCS) wallet location
- The connection uses mutual TLS
- There is more than one private key containing certificates in the MY/CurrentUser MCS

If `AllowCertificateSelectionUI` is false and configuration-based certificate selection is not set (e.g. thumbprint is not set), then ODP.NET will choose the first certificate with a private key in the MCS.

If enabling this property, set its value to true prior to opening the ODP.NET connection.

ODP.NET will filter out trusted certificates (public key only) from the end user display to simplify their selection process.

After a certificate is selected in the user interface, it is cached via a key. The key value consists of the connection string, database server host, and port number. The cache can be cleared by calling the `ClearAllPools()` method.

To specify the certificate via a thumbprint instead of a graphical interface, use the `OracleConfiguration` or `OracleConnection.SSLCertificateThumbprint` property.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

AutoCommit

This property specifies whether to commit each SQL statement implicitly or only until an explicit commit occurs.

Declaration

```
// C#  
public bool AutoCommit {get; set;}
```

Property Value

Bool

Exception

`System.ObjectDisposedException` – This property cannot be accessed after the `OracleConnection` object is already disposed.

Remarks

By default, this property is set to true, which commits each SQL statement implicitly.

This property can be set before or after the connection opens.

If the `AutoCommit` property is toggled from false to true, the current transaction is committed.

When an explicit transaction is created, this property will be set to false. Once the explicit transaction is committed or rolled back, the property value reverts to the originally set `AutoCommit` value.

If a local transaction is already started implicitly, invoking `BeginTransaction()` will inherit the transaction that was implicitly started.

If the transaction is created explicitly using `BeginTransaction()`, the transaction can be operated on either through the `OracleConnection` methods or `OracleTransaction` methods. But once the transaction is over via a `Commit()` or a `Rollback()` invoked on either object, the `OracleTransaction` cannot be used henceforth.

If the `OracleConnection` object is closed while an implicit transaction is underway, the transaction is rolled back. After the `AutoCommit` property is set, that setting is honored for all subsequent `Open()` calls on that specific `OracleConnection` object until the `AutoCommit` value is modified.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleConnection Class](#)
- [OracleConnection Members](#)

AutoProxy

This property enables using a HTTPS proxy server for the ODP.NET connection.

Declaration

```
// C#  
public bool AutoProxy {get; set;}
```

Property Type

`System.Boolean`

Remarks

Default is null. When set to false or null, `OracleConnection` `HttpsProxy` and `HttpsProxyPort` properties are used for the HTTPS proxy server configuration.

If set to true, the default or system proxy server is used as the HTTPS proxy server. `HttpsProxy` and `HttpsProxyPort` are then ignored.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleConnection Class](#)
- [OracleConnection Members](#)

BindByName

This property specifies whether the binding method used for the parameter collection is by name or by position on a particular `OracleConnection` object.

Declaration

```
// C#  
public bool BindByName { get; set;}
```

Property Type

`System.Boolean`

Property Value

Returns `true` if the parameters are bound by name; returns `false` if the parameters are bound by position.

Exceptions

`System.ObjectDisposedException` - This property cannot be accessed after the `OracleConnection` object is already disposed.

Remarks

The default value is `false`.

`BindByName` is ignored under the following conditions:

- The value of the `XmlCommandType` property is `Insert`, `Update`, or `Delete`.
- The value of the `XmlCommandType` property is `Query`, but there are no parameters set on the `OracleCommand`.

If the `OracleCommand` `XmlCommandType` property is `OracleXmlCommandType.Query` and any parameters are set on the `OracleCommand`, then its `BindByName` property must be set to `true`. Otherwise, the following `OracleCommand` methods throw an `InvalidOperationException`.

- `ExecuteNonQuery`
- `ExecuteXmlReader`
- `ExecuteStream`
- `ExecuteToStream`

The `BindByName` property can be configured on `OracleCommand`, `OracleConnection`, and `OracleConfiguration` objects. By default, `OracleConnection` `BindByName` property value inherits `OracleConfiguration` `BindByName` property value at construction time; `OracleCommand` `BindByName` property value inherits `OracleConnection` `BindByName` property value at construction time. The value of `OracleCommand` `BindByName` will be used to determine the binding method during command execution.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

ChunkMigrationConnectionTimeout

In sharding, a connection request to a specific shard can occur while data is migrating from the original shard to a new one. This setting will pause the connection request until the migration completes or fails to migrate in the specified time, then connect to the correct shard with the data requested.

Declaration

```
// C#  
public int ChunkMigrationConnectionTimeout {get;set;}
```

Property Value

The maximum time, in seconds, for a pooled connection request to pause waiting for a chunk migration or split partitionset for a particular chunk.

Remarks

If a specific shard connection request occurs while data is migrating to a new shard or being split and moved to a new shard through a split partitionset operation, then this setting pauses the connection request until the data migration completes or fails in the specified time. It then connects to the correct shard with the data requested.

Default is 120 seconds. This setting requires connection pooling to be enabled.

ODP.NET reads the chunk migration connection timeout setting only upon pool creation. A new timeout setting will not be used until a new pool is created.

This setting is in effect if

- write access is required and a chunk is read-only, or
- during a chunk migration or split partitionset, the chunk becomes unavailable to read.

If the connection requires only read-only access, then this setting will not be used unless the chunk becomes unavailable.

If the connection requires write access and the chunk does not become writable or available before the chunk migration connection timeout expires, then a new connection will be created. If the connection cannot be created, then an `OracleException` will occur.

During chunk migration or split partitionset, an ODP.NET connection request obeys only the chunk migration connection timeout. All other connection timeout values, such as `OracleConnection.ConnectionTimeout`, are ignored.

Oracle recommends setting the ODP.NET chunk migration connection timeout to a value greater than the chunk move server timeout. One way to set this server timeout is using the Global Data Services Control Utility (GDSCCTL) `move chunk` command's `timeout` option. This server-side timeout only applies to chunk migrations, not split partitionset.

The `OracleConnection ChunkMigrationConnectionTimeout` property inherits from the `OracleConfiguration ChunkMigrationConnectionTimeout` property. The `OracleConnection` property can be separately set to override the property set in the `OracleConfiguration` class.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

ClientId

This property specifies the client identifier for the connection.

Declaration

```
// C#  
public string ClientId {set;}
```

Property Value

The string to be used as the client identifier.

Remarks

The default value is null.

Using the `ClientId` property allows the application to set the client identifier in the application context for a given `OracleConnection` object.

For unmanaged ODP.NET with `CPVersion = 1.0`, the `ClientId` property is reset to null upon the next server roundtrip following a `Close` or `Dispose` method call on the `OracleConnection` object. For all other ODP.NET provider types and `CPVersion` settings, `ClientId` property is reset to null when the `Close` or `Dispose` method is called on the `OracleConnection` object.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- ["Client Identifier and End-to-End Tracing"](#)
- *Oracle Database Security Guide*

ClientInfo

This property specifies the client information for the connection.

Declaration

```
// C#  
public string ClientInfo {set;}
```

Property Value

The string to be used as the client information.

Remarks

The default value is null.

Using the ClientInfo property allows the application to set the client information in the application context for a given OracleConnection object.

For unmanaged ODP.NET with CPVersion = 1.0, the ClientInfo property is reset to null upon the next server roundtrip following a Close or Dispose method call on the OracleConnection object. For all other ODP.NET provider types and CPVersion settings, ClientInfo property is reset to null when the Close or Dispose method is called on the OracleConnection object.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- ["Client Identifier and End-to-End Tracing"](#)
- *Oracle Database Security Guide*

ClientTimeZoneFileVersion

This property returns the highest time zone file version in the TimeZoneFileLocation directory and embedded in ODP.NET.

Declaration

```
// C#  
public string ClientTimeZoneFileVersion {get;}
```

Property Type

int

Exception

InvalidOperationException - The connection is not in an open state.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

CommandTimeout

This property specifies the minimum number of seconds that the command is allowed to execute before terminating with an exception on a particular `OracleConnection` object.

Declaration

```
// C#  
public int CommandTimeout { get; set;}
```

Property Type

`System.Int32`

Exceptions

`InvalidArgument` - The specified value is less than 0 or greater than `System.Int32.MaxValue`.

`System.ObjectDisposedException` - This property cannot be accessed after the `OracleConnection` object is already disposed.

Remarks

Default is 0 (seconds), which enforces no time limit.

When the specified timeout value expires before a command execution finishes, ODP.NET will begin the process of canceling the command. If cancellation is successful, then an exception is thrown with the message: ORA-01013: user requested cancel of current operation. Other possible exceptions thrown after a command timeout expiration occurs include ORA-00936 and ORA-00604. If the command is executed in time without any errors, then no exceptions are thrown.

ODP.NET does not guarantee a command will be cancelled at the exact moment the timeout value is reached. The timeout value is the minimum time to allow a command to complete before the cancellation process begins. It is possible for commands to complete after the timeout value expires.

In a situation where multiple `OracleCommand` objects use the same connection, the timeout expiration on one of the `OracleCommand` objects may terminate any of the executions on the single connection. To make the timeout expiration of a `OracleCommand` cancel only its own command execution, simply use one `OracleCommand` for each connection if that `OracleCommand` sets the `CommandTimeout` property to a value greater than 0.

The `CommandTimeout` property can be configured on `OracleCommand`, `OracleConnection`, and `OracleConfiguration` objects. By default, `OracleConnection` `CommandTimeout` property value inherits `OracleConfiguration` `CommandTimeout` property value at construction time; `OracleCommand` `CommandTimeout` property value inherits `OracleConnection` `CommandTimeout` property value at construction time. The value of `OracleCommand` `CommandTimeout` will be used during command execution.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

ConnectionIdPrefix

This property is used to identify a particular application's connections by appending its string to the connection identifier. The identifier and prefix can be used to uniquely identify the application connections for tracing and diagnosis purposes.

Declaration

```
// C#  
public string ConnectionIdPrefix { set; }
```

Property Value

System.String

Remarks

The property value is an 8-byte identifier limited to alphanumeric characters and underscore (_).

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

ConnectionString

This property specifies connection information or URL used to connect to an Oracle database.

Declaration

```
// C#  
public override string ConnectionString { get; set; }
```

Property Value

If the connection string is supplied through the constructor, this property is set to that string.

Implements

IDbConnection

Exceptions

`ArgumentException` - An invalid syntax is specified for the connection string.

`InvalidOperationException` - `ConnectionString` is being set while the connection is open.

Remarks

The default value is an empty string.

`ConnectionString` must be a connection string URL or a string of attribute name and value pairings, separated by a semi-colon, for example:

```
"User Id=scott;password=tiger;data source=oracle"
```

Connection string URLs are used with centralized configuration providers.

If the `ConnectionString` is not in a proper format, an exception is thrown. All spaces are ignored unless they are within double quotes.

When the `ConnectionString` property is set, the `OracleConnection` object immediately parses the string for errors. An `ArgumentException` is thrown if the `ConnectionString` contains invalid attributes or invalid values. Attribute values for User Id, Password, Proxy User Id, Proxy Password, and Data Source (if provided) are not validated until the `Open` method is called.

The connection must be closed to set the `ConnectionString` property. When the `ConnectionString` property is reset, all previously set values are reinitialized to their default values before the new values are applied.

Starting with ODP.NET 11.1, password and proxy password connection string attribute values are accepted as case-sensitive strings. Thus, they are passed to the database for authentication in the case provided in the connection string. Therefore, if the database is configured to support case-sensitive passwords, passwords must be passed in the correct case.

If a connection string attribute is set more than once, the last setting takes effect and no exceptions are thrown.

Boolean connection string attributes can be set to either `true`, `false`, `yes`, or `no`.

Supported Connection String Attributes

[Table 6-51](#) lists the supported connection string attributes.

Table 6-51 Supported Connection String Attributes

Connection String Attribute	Description	Default Value
Application Continuity	Enables database requests to automatically replay transactional or non-transactional operations in a non-disruptive and rapid manner in the event of a severed database session, which results in a recoverable error.	true

Table 6-51 (Cont.) Supported Connection String Attributes

Connection String Attribute	Description	Default Value
Connection Lifetime	Minimum life time (in seconds) of the connection. This attribute specifies the lifetime of the connection in seconds. Before the <code>Connection</code> is placed back into the pool upon a <code>Close()</code> or <code>Dispose()</code> call, the lifetime of the connection is checked. If the lifetime of the connection exceeds this property value and the number of connections will not fall below <code>Min Pool Size</code>, then the connection is destroyed. If this property value is 0, then the connection lifetime is never checked.	0
Connection Timeout	The time to wait (in seconds) for a new connection or an idle connection from the connection pool before a connection time out error can occur. This attribute specifies the minimum amount of time (in seconds) that the <code>Open()</code> method must take to obtain a pooled connection before it terminates the request. This value comes into effect only if no free connection is available from the connection pool and the <code>Max Pool Size</code> is reached. If a free connection is not available within the specified time, an exception is thrown. <code>Connection Timeout</code> does not limit the time required to open new connections. This attribute value takes effect for pooled connection requests and not for new connection requests.	15
Data Source	Oracle Net Services Name, Connect Descriptor, or an easy connect naming that identifies the database to which to connect.	empty string
DBA Privilege	Administrative privileges. This connection string attribute accepts <code>SYSASM</code>, <code>SYSBACKUP</code>, <code>SYSDBA</code>, <code>SYSDG</code>, <code>SYSKM</code>, <code>SYSOPER</code>, and <code>SYSRAC</code> as the attribute value. It is case-insensitive.	empty string
Decr Pool Size	Number of connections that are closed when an excessive amount of established connections are unused. This connection string attribute controls the maximum number of unused connections that are closed when the pool regulator makes periodic checks. The regulator thread is spawned every 3 minutes and closes up to <code>Decr Pool Size</code> amount of pooled connections if they are not used. The pool regulator never takes the total number of connections below the <code>Min Pool Size</code> by closing pooled connections.	1

Table 6-51 (Cont.) Supported Connection String Attributes

Connection String Attribute	Description	Default Value
Enlist	Controls the enlistment behavior and capabilities of a connection in context of COM+ transactions or System.Transactions. If this attribute is set to <code>true</code> , the connection is automatically enlisted in the thread's transaction context. If this attribute is <code>false</code> , no enlistments are made. If this attribute is set to <code>dynamic</code> , applications can dynamically enlist in distributed transactions. This attribute can be set to <code>true</code> , <code>false</code> , <code>yes</code> , <code>no</code> , or <code>dynamic</code> .	<code>true</code>
HA Events	Enables ODP.NET connection pool to proactively remove connections from the pool when an Oracle database service, service member, or node goes down. This feature can be used with Global Data Services, including Oracle RAC, Data Guard, GoldenGate, and single instance deployments. " <code>pooling=true</code> " must also be set This attribute can be set to <code>true</code> , <code>false</code> , <code>yes</code> , or <code>no</code> .	<code>true</code>
Load Balancing	Enables ODP.NET connection pool to balance work requests across Oracle database instances based on the load balancing advisory and service goal. This feature can be used with Global Data Services, including Oracle RAC, Active Data Guard, and GoldenGate. " <code>pooling=true</code> " must also be set. This attribute can be set to <code>true</code> , <code>false</code> , <code>yes</code> , or <code>no</code> .	<code>true</code>
Incr Pool Size	Number of new connections to be created when all connections in the pool are in use. This connection string attribute determines the number of new connections that are established when a pooled connection is requested, but no unused connections are available and Max Pool Size is not reached. If new connections have been created for a pool, the regulator thread skips a cycle and does not have an opportunity to close any connections for 6 minutes. Note, however, that some connections can be still be closed during this time if their lifetime has been exceeded.	5
Max Pool Size	Maximum number of connections in a pool. This attribute specifies the maximum number of connections allowed in the particular pool used by that OracleConnection. Simply changing this attribute in the connection string does not change the Max Pool Size restriction on a currently existing pool. Doing so simply creates a new pool with a different Max Pool Size restriction. This attribute must be set to a value greater than the Min Pool Size. This value is ignored unless Pooling is turned on.	100

Table 6-51 (Cont.) Supported Connection String Attributes

Connection String Attribute	Description	Default Value
Metadata Pooling	Caches metadata information. This attribute indicates whether or not metadata information for executed queries are cached for improved performance.	True
Min Pool Size	Minimum number of connections in a pool. This attribute specifies the minimum number of connections to be maintained by the pool during its entire lifetime. Simply changing this attribute in the connection string does not change the Min Pool Size restriction on a currently existing pool. Doing so simply creates a new pool with a different Min Pool Size restriction. This value is ignored unless Pooling is turned on.	1
Password	Password for the user specified by User Id. This attribute specifies an Oracle user's password. Password is case-sensitive by default for Oracle Database 11g release 1 (11.1) and later.	empty string
Persist Security Info	Retrieval of the password in the connection string. If this attribute is set to false, the Password value setting is not returned when the application requests the ConnectionString after the connection is successfully opened by the Open() method. This attribute can be set to either true, false, yes, or no.	false
Pooling	Connection pooling. This attribute specifies whether or not connection pooling is to be used. Pools are created using an attribute value matching algorithm. This means that connection strings which only differ in the number of spaces in the connection string use the same pool. If two connection strings are identical except that one sets an attribute to a default value while the other does not set that attribute, both requests obtain connections from the same pool. This attribute can be set to either true, false, yes, or no.	true
Proxy User Id	User name of the proxy user. This connection string attribute specifies the middle-tier user, or the proxy user, who establishes a connection on behalf of a client user specified by the User Id attribute. ODP.NET attempts to establish a proxy connection if either the Proxy User Id or the Proxy Password attribute is set to a non-empty string. For the proxy user to connect to an Oracle database using operating system authentication, the Proxy User Id must be set to "/". The Proxy Password is ignored in this case. The User Id cannot be set to "/" when establishing proxy connections. The case of this attribute value is preserved.	empty string

Table 6-51 (Cont.) Supported Connection String Attributes

Connection String Attribute	Description	Default Value
Proxy Password	Password of the proxy user. This connection string attribute specifies the password of the middle-tier user or the proxy user. This user establishes a connection on behalf of a client user specified by the User Id attribute. ODP.NET attempts to establish a proxy connection if either the Proxy User Id or the Proxy Password attribute is set to a non-empty string. The case of this attribute value is preserved if it is surrounded by double quotes.	empty string
Statement Cache Purge	Statement cache purged when the connection goes back to the pool. If statement caching is enabled, setting this attribute to <code>true</code> purges the Statement Cache when the connection goes back to the pool.	false
Statement Cache Size	Statement cache enabled and cache size set size, that is, the maximum number of statements that can be cached. A value greater than zero enables statement caching and sets the cache size to itself. This value should not be greater than the value of the <code>OPEN_CURSORS</code> parameter set in the <code>init.ora</code> database configuration file.	0
Self Tuning	Enables or disables self-tuning for the connection. If self-tuning is enabled, then the <code>StatementCacheSize</code> settings in the registry, configuration files, and connection string are ignored. If self-tuning is disabled, then a <code>StatementCacheSize</code> value of 0 is used unless <code>StatementCachSize</code> is specified in the registry, configuration file, or connection string.	true
Tns_Admin	Directory where ODP.NET can find its <code>sqlnet.ora</code> and <code>tnsnames.ora</code> configuration files. <i>Not available in ODP.NET, Unmanaged Driver</i>	empty string
Token_Auth	This attribute specifies the access token authentication type. Possible values are: AZURE_INTERACTIVE, AZURE_SERVICE_PRINCIPAL, AZURE_MANAGED_IDENTITY, AZURE_DEVICE_CODE, AZURE_DEFAULT, OCITOKEN, OAUTH, or DISABLED. Only supported for managed ODP.NET and ODP.NET Core.	DISABLED

Table 6-51 (Cont.) Supported Connection String Attributes

Connection String Attribute	Description	Default Value
Token_Location	This attribute is the file-based token location. The value can be a directory where a file named "token" is or it can be the file's full path specification. Only supported for managed ODP.NET and ODP.NET Core.	Varies depending on token authentication type
User Id	Oracle user name. This attribute specifies the Oracle user name. The case of this attribute value is preserved if it is surrounded by double quotes. For the user to connect to an Oracle database using operating system authentication, set the User Id to "/". Any Password attribute setting is ignored in this case.	empty string
Validate Connection	Validation of connections coming from the pool. Validation causes a round-trip to the database for each connection. Therefore, it should only be used when necessary.	false
Wallet_Location	ODP.NET wallet directory <i>Not available in ODP.NET, Unmanaged Driver</i>	empty string
Wallet_Password	Stores PKCS #12 wallet password <i>Not available in ODP.NET, Unmanaged Driver</i>	empty string

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- [Centralized Configuration Providers for Deployments](#) for more information on using URLs

ConnectionTimeout

This property indicates the minimum amount of time that the Open method can take to obtain a pooled connection before the request is terminated.

Declaration

```
// C#
public override int ConnectionTimeout {get;}
```

Property Value

The minimum time allowed for a pooled connection request, in seconds.

Implements

IDbConnection

Remarks

This property indicates the connection timeout that has been set using the `ConnectionString` attribute `Connection Timeout`.

This property is read-only.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

ConnectionType

This property enables an ODP.NET application to determine whether a particular connection object is associated with an Oracle database connection, a TimesTen database connection, or no physical connection at all.

Declaration

```
// C#  
public OracleConnectionType ConnectionType {get;}
```

Property Value

The `OracleConnectionType` that this connection object is associated with.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- [OracleConnectionType Enumeration](#)

Credential

This property is used to get or set an `OracleCredential` object on the connection.

Declaration

```
// C#  
public OracleCredential Credential { get; set; }
```

Property Value

An `OracleCredential` object to be used for opening connection.

Exceptions

An `InvalidOperationException` will be raised in following scenarios:

- If connection is already open.
- If connection string has already been set on this connection and it contains any of user id, password, DBA Privilege, proxy user id or proxy password.
- The `AccessToken` property is set.
- The `TokenLocation` properties is set. This is applicable for managed ODP.NET and ODP.NET Core only.
- `TokenAuthentication` is set to something other than disabled. This is applicable for managed ODP.NET and ODP.NET Core only.

Remarks

To obtain the `OracleCredential` object through the property getter, "persist security info" must be set to true in the connection string.

If "Persist security info" is set to false in the Connection String, then the `OracleCredential` object is not returned through the property getter after the password has been validated.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- [OracleConnectionType Enumeration](#)

Database

This property is not supported.

Declaration

```
// C#  
public override string Database {get;}
```

Property Value

A string.

Implements

`IDbConnection.Database`

Remarks

This property is not supported. It always returns an empty string.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

DatabaseCharset

This property specifies the CHAR database character set on the connection and is used to optimize connection establishment.

Declaration

```
// C#  
public OracleDatabaseCharset DatabaseCharset {get; set;}
```

Remarks

The default value is `OracleDatabaseCharset.AL32UTF8`.

When the application provides the actual CHAR database character set to connect with, the connection can be established in an optimized way with fewer roundtrips. If the property contains a non-matching character set value, connection establishment follows a traditional, less optimized process.

When retrieving the database character set value, the connection must be open. If it is not open, an exception occurs.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- [OracleDatabaseCharset Enumeration](#)

DatabaseNCharset

This property specifies the NCHAR database character set on the connection and is used to optimize connection establishment.

Declaration

```
// C#  
public OracleDatabaseNCharset DatabaseNCharset {get; set;}
```

Remarks

The default value is `OracleDatabaseNCharset.AL16UTF16`.

When the application provides the actual NCHAR database character set to connect with, the connection can be established in an optimized way with fewer roundtrips. If the property contains a non-matching character set value, connection establishment follows a traditional, less optimized process.

When retrieving the database character set value, the connection must be open. If it is not open, an exception occurs.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- [OracleDatabaseNCharset Enumeration](#)

DatabaseDomainName

This property specifies the name of the database domain that this connection is connected to.

Declaration

```
// C#  
public string DatabaseDomainName {get;}
```

Property Value

The database domain that this connection is connected to.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

DatabaseEditionName

DatabaseEditionName property sets or gets the edition name for the OracleConnection object.

Declaration

```
// C#  
public string DatabaseEditionName{get;set;}
```

Property Value

A string to be used as the edition name for the OracleConnection object.

Exceptions

- `InvalidOperationException()` - if the connection is already open.

- `InvalidOperationException()` - If `CPVersion` is set to 1.0 when the connection is opened. `DatabaseEditionName` property is supported only with `CPVersion` of 2.0 or higher.

Remarks

Property must be set before opening a connection or after closing a connection.

The edition name for the connection to use can be set in one of three places: `DatabaseEditionName` in the application code, `Edition` in the .NET configuration file, or `Edition` in the Windows Registry. The .NET configuration value overrides the Registry value and the `DatabaseEditionName` property value overrides the .NET configuration value.

`DatabaseEditionName` property value is case sensitive.

Only supported for .NET Framework 4 and higher.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

DatabaseMaxIdentifierLength

This property specifies the maximum identifier length the connection supports.

Declaration

```
// C#  
public int DatabaseMaxIdentifierLength {get;}
```

Remarks

The connection must be open when using this property. If the connection is closed, an exception occurs.

The Oracle Database maximum identifier length varies depending on the version. Older Oracle databases supported a 30 byte maximum and newer databases support a 128 byte maximum.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- [OracleDatabaseNCharset Enumeration](#)

DatabaseName

This property specifies the name of the database that this connection is connected to.

Declaration

```
// C#  
public string DatabaseName {get;}
```

Property Value

The database that this connection is connected to.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

DatabaseTimeZoneFileVersion

This property returns the database's time zone file version.

Declaration

```
// C#  
public string DatabaseTimeZoneFileVersion {get;}
```

Property Type

int

Remarks

An alternative method to check the Oracle database's time zone file version is to run the following query:

```
SELECT * FROM V$TIMEZONE_FILE ;
```

Exception

InvalidOperationException - The connection is not in an open state.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

DataSource

This property specifies the Oracle Net Services Name, Connect Descriptor, or an easy connect naming that identifies the database to which to connect

Declaration

```
// C#  
public override string DataSource {get;}
```

Property Value

Oracle Net Services Name, Connect Descriptor, or an easy connect naming that identifies the database to which to connect.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

DRCPConnectionClass

This property specifies a logical name that identifies the DRCP connection pool that the ODP.NET connection will use.

Declaration

```
// C#  
public string DRCPConnectionClass {get; set;}
```

Property Value

The string to be used that uniquely identifies the DRCP server side connection pool for the ODP.NET application to use.

Exceptions

- `InvalidOperationException()` - if `CPVersion=1.0` and the `DRCPConnectionClass` is set to a non-null / non-empty string. Applies to unmanaged ODP.NET only.
- `InvalidOperationException()` - if the `DRCPConnectionClass` is set to a non-null / non-empty string after opening a connection.
- `InvalidOperationException()` - if DRCP is not enabled and the `DRCPConnectionClass` is set to a non-null / non-empty string.

Remarks

A different client-side connection pool will be created for each unique `DRCPConnectionClass` property value. Each of these pools will use the same DRCP.

The default value is null. The character limit is 1024 minus the number of characters in the user id.

If this property is used, it must be set prior to opening the connection.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

DRCP Purity

Session purity specifies whether an application can reuse a pooled session (Pooled) or must use a new session (New).

Declaration

```
// C#  
public OracleDRCP Purity {get; set;}
```

Property Value

OracleDRCP Purity Enumeration values.

Exceptions

- `InvalidOperationException()` - if `CPVersion=1.0` and the `DRCP Purity` is set to a non-null / non-empty string. Applies to unmanaged ODP.NET only.
- `InvalidOperationException()` - if the `DRCP Purity` is set to a non-null / non-empty string after opening a connection.
- `InvalidOperationException()` - if DRCP is not enabled and the `DRCP Purity` is set to a non-null / non-empty string.

Remarks

The default value is Pooled.

If set to New, then ODP.NET will dispense a new DRCP connection, rather than try to find an existing connection from the DRCP. In general, this property is used by applications that want a connection without any preexisting session state set.

If set to Pooled, then ODP.NET will first attempt to use a preexisting connection, that already exists in the DRCP pool.

Using this property will not cause a new DRCP to be created, just a new connection from an existing DRCP.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- [OracleDRCP Purity Enumeration](#)

FetchSize

This property specifies the total memory size, in bytes, that the provider allocates to cache data fetched in one database round-trip, used by `OracleDataReader`'s internal cache to store result set data.

Declaration

```
// C#  
public long FetchSize { get; set;}
```

Property Type

`System.Int64` (long)

Exceptions

`InvalidArgument` - The specified value is less than 0 or greater than `System.Int64.MaxValue`.

`System.ObjectDisposedException` - This property cannot be accessed after the `OracleConnection` object is already disposed.

Remarks

The default value is 131072.

The `FetchSize` property can be configured on `OracleDataReader`, `OracleCommand`, `OracleConnection`, and `OracleConfiguration` objects. By default, `OracleConnection` `FetchSize` property value inherits `OracleConfiguration` `FetchSize` property value at construction time; `OracleCommand` `FetchSize` property value inherits `OracleConnection` `FetchSize` property value at construction time; `OracleDataReader` `FetchSize` property value inherits `OracleCommand` `FetchSize` property value at a command execution. The value of `OracleDataReader` `FetchSize` determines the amount of data the `OracleDataReader` fetches into its internal cache for each database round-trip.

If the `OracleCommand.XmlCommandType` property is set to any value other than `None`, this property is ignored.

The `OracleCommand` `RowSize` and `OracleCommand` `FetchSize` properties handle UDT and XMLType data differently than other scalar data types. Because only a reference to the UDT and XMLType data is stored in the ODP.NET's internal cache, the `OracleCommand` `RowSize` property accounts for only the memory needed for the reference (which is very small) and not the actual size of the UDT and XMLType data. Thus, applications can inadvertently fetch a large number of UDT or XMLType instances from the database in a single database round-trip. This is because the actual size of UDT and XMLType data do not count against the `FetchSize`, and it would require numerous UDT and XMLType references to fill up the default cache size of 131072 bytes. Therefore, when fetching UDT or XMLType data, the `FetchSize` property must be appropriately configured to control the number of UDT and XMLType instances that are to be fetched, rather than the amount of the actual UDT and XMLType data to be fetched.

Note

For LOB and LONG data types, only the sizes specified in the `OracleCommand InitialLOBFetchSize` and `OracleCommand InitialLONGFetchSize` properties are accounted for by the `RowSize` property in addition to the metadata and reference information that is maintained by the cache for each LOB in the select list.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

HostName

This property specifies the name of the host that this connection is connected to.

Declaration

```
// C#  
public string HostName {get;}
```

Property Value

The host that this connection is connected to.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

HttpsProxy

This property specifies the HTTPS proxy server.

Declaration

```
// C#  
public string HttpsProxy {get; set;}
```

Property Type

System.String

Remarks

The value must be set to a valid host name or IP address of a HTTPS proxy server. This property has no default value. An invalid host name or IP address throws an exception.

This property overrides all other HTTPS_PROXY settings, including the full connect descriptor and Easy Connect Plus configuration.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

HttpsProxyPort

This property specifies the HTTPS proxy server port number.

Declaration

```
// C#  
public static int HttpsProxyPort {get; set;}
```

Remarks

The default value is 80. The value must be between 0 and 65535.

An invalid specified port throws an exception.

This property overrides the HTTPS_PROXY_PORT setting, including the full connect descriptor and Easy Connect Plus configuration.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

InstanceName

This property specifies the name of the instance that this connection is connected to.

Declaration

```
// C#  
public string InstanceName {get;}
```

Property Value

The instance that this connection is connected to.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

IsInTransaction

This property specifies if the current application is in a transaction.

Declaration

```
// C#  
public bool IsInTransaction {get;}
```

Property Value

Boolean

Remarks

This property will be true if the connection is in a local implicit or explicit transaction, sessionless transaction, or a distributed transaction.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

IsNNEChecksummed

This property indicates if the connection uses Oracle Database Native Network Encryption (NNE) checksumming.

Declaration

```
// C#  
public bool IsNNEChecksummed { get; }
```

Property Type

System.Boolean

Exceptions

InvalidOperationException - This exception is thrown when the connection is not open.

Remarks

This property is true if the connection uses NNE checksumming; Otherwise, it is false.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

IsNNEncrypted

This property indicates if the connection uses Oracle Database Native Network Encryption (NNE).

Declaration

```
// C#  
public bool IsNNEncrypted { get; }
```

Property Type

System.Boolean

Exceptions

InvalidOperationException - This exception is thrown when the connection is not open.

Remarks

This property is true if the connection uses NNE; Otherwise, it is false.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

IsTLSEncrypted

This property indicates if the connection uses TCPS, SSL, or TLS.

Declaration

```
// C#  
public bool IsTLSEncrypted { get; }
```

Property Type

System.Boolean

Exceptions

InvalidOperationException - This exception is thrown when the connection is not open.

Remarks

This property is true if the connection uses TCPS, SSL, or TLS; Otherwise, it is false.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

KeepAlive

This property specifies whether to keep an idle connection alive.

Declaration

```
// C#  
public Boolean KeepAlive { get; set; }
```

Property Type

System.Boolean

Remarks

Turn KeepAlive on to prevent an idle TCP connection from being killed, such as a by a firewall. KeepAlive will keep the connection alive by periodically sending a probe packet with no data in it and the ACK flag turned on.

This setting is used in conjunction with KeepAliveInterval and KeepAliveTime.

The KeepAlive default is false.

Starting in ODP.NET Core 19.10, KeepAlive is supported on non-Windows operating systems, such as Oracle Linux. For these non-Windows operating systems, .NET Core 3 or higher is required.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

KeepAliveInterval

This property specifies the time interval after an unacknowledged probe before a new probe is sent.

Declaration

```
// C#  
public int KeepAliveInterval { get; set; }
```

Property Type

System.Int32

Remarks

In seconds. The interval default is 6 seconds between unacknowledged probes.

When keepalive probes are sent, they are normally acknowledged by the remote TCP. If the connection remains idle, but TCP does see the probe acknowledgements, then TCP will send the probes at a rate of once every KeepAliveTime duration.

When these probes are not acknowledged, the probes are then sent at intervals set by KeepAliveInterval. Probes will continue to be sent at the KeepAliveInterval until the server acknowledges the probe, independent connection activity occurs, or 10 consecutive unacknowledged probes have been sent.

After a probe is acknowledged, future probes will be sent again at the KeepAliveTime interval.

If 10 probes are sent without acknowledgement, the connection is deemed EOF/Reset.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

KeepAliveTime

This property specifies the idle time before a keepalive probe is sent.

Declaration

```
// C#  
public int KeepAliveTime { get; set; }
```

Property Type

System.Int32

Remarks

In seconds. The default is 60 seconds.

KeepAliveTime specifies the idle time to wait until sending a probe to verify the connection remains active and to keep that connection active. If the remote TCP sends back an acknowledgement of receiving the probe, then no further probe is sent until another KeepAliveTime idle period has elapsed.

Whenever independent connection activity occurs, KeepAliveTime is reset. When that activity ends, the KeepAliveTime starts its countdown to sending a probe.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

MaxStatementCacheSize

This property specifies the maximum number of statements that can be cached on a particular OracleConnection object when self-tuning is enabled.

Declaration

```
// C#  
public int MaxStatementCacheSize { get; set;}
```

Property Type

System.Int32

Exceptions

InvalidArgument - The specified value is less than 0 or greater than System.Int32.MaxValue .

System.ObjectDisposedException - This property cannot be accessed after the OracleConnection object is already disposed.

Remarks

The default value is 100.

The MaxStatementCacheSize property can be configured on OracleConnection and OracleConfiguration objects. By default, OracleConnection MaxStatementCacheSize property value inherits OracleConfiguration MaxStatementCacheSize property value at construction time. The value of OracleConnection MaxStatementCacheSize will be used during the connection.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- [Self-Tuning Statement Caching](#)

ModuleName

This property specifies the module name for the connection.

Declaration

```
// C#  
public string ModuleName {set;}
```

Property Value

The string to be used as the module name.

Remarks

The default value is null.

Using the ModuleName property allows the application to set the module name in the application context for a given OracleConnection object.

For unmanaged ODP.NET with CPVersion = 1.0, the ModuleName property is reset to null upon the next server roundtrip following a Close or Dispose method call on the OracleConnection object. For all other ODP.NET provider types and CPVersion settings, ModuleName property is reset to null when the Close or Dispose method is called on the OracleConnection object.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- ["Client Identifier and End-to-End Tracing"](#)
- *Oracle Database Security Guide*

NamesDirectoryPath

This property specifies the naming method order used for client name resolution lookups.

Declaration

```
// C#  
public string NamesDirectoryPath { get; set; }
```

Property Type

System.String

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

OciCompartment

This property gets or sets the Oracle Cloud Identifier (OCID) of the target database's compartment.

Declaration

```
// C#  
public string OciCompartment { get; set; }
```

Property Type

System.String

Remarks

This property is optional by default. This property is mandatory if the `OciDatabase` property is set.

If `OciCompartment` is not set, then ODP.NET requests access to all the cloud tenancy's databases identified in the `OciTenancy` property.

There is no default value for this property.

The value of this property will be part of pool manager identification so different value for this property will result in different connection pools.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

OciDatabase

This property gets or sets the target database's Oracle Cloud Identifier (OCID).

Declaration

```
// C#  
public string OciDatabase { get; set; }
```

Property Type

System.String

Remarks

This property is optional.

If it is not set, then ODP.NET requests access to all the cloud compartment's databases identified in the `OciCompartment` property. If `OciCompartment` property is also not set, then ODP.NET requests access to all the tenancy's databases specified in the `OciTenancy` property.

There is no default value for this property.

The value of this property will be part of pool manager identification so different value for this property will result in different connection pools.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

OciIamUrl

This property gets or sets the URL end point for getting the database token.

Declaration

```
// C#  
public string OciIamUrl { get; set; }
```

Property Type

System.String

Remarks

This is a mandatory property for IAM alternate password use. If not set in that usage scenario, ODP.NET raises an error.

The property value is part of the connection pool manager identification. Different property values result in different connection pools.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

OciTenacy

This property gets or sets the Oracle Cloud Identifier (OCID) of the IAM user's cloud tenancy to retrieve the database token.

Declaration

```
// C#  
public string OciTenacy { get; set; }
```

Property Type

System.String

Remarks

This is a mandatory property for IAM alternate password use. If not set in that usage scenario, ODP.NET raises an error.

The property value is part of the connection pool manager identification. Different property values result in different connection pools.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleConnection Class](#)
- [OracleConnection Members](#)

PasswordAuthentication

This property sets the connection PASSWORD_AUTH parameter value.

Declaration

```
// C#  
public OraclePasswordAuth PasswordAuthentication { get; set; }
```

Property Type

An OraclePasswordAuth enumeration

Remarks

The possible parameter values are OciToken, AzureToken, and PasswordVerifier. The default value is PasswordVerifier, which indicates the typical database username and password can be used for authentication. AzureToken is only available with ODP.NET Core and managed ODP.NET.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleConnection Class](#)
- [OracleConnection Members](#)

PDBName

This property is used to set and get the name of the pluggable database to which the session is connected to.

Declaration

```
// C#  
public string PDBName { get; set; }
```

Property Value

PDBName returns a string specifying the pluggable database name for the connection.

Exceptions

- `InvalidOperationException()` - if the connection is already open.
- `InvalidOperationException()` - if the `CPVersion` is set to 1.0 and the `PDBName` property is set. Applicable only to ODP.NET, Unmanaged Driver.
- `InvalidOperationException()` - If `PDBName` value is retrieved when `CPVersion` is set to 1.0. Applicable only to ODP.NET, Unmanaged Driver.

Remarks

Pluggable databases are available in Oracle Database 12c and higher. The `ServiceName` and `PDBName` properties can be used to connect to the specified Service that is running on a particular PDB.

Property must be set before opening the connection so that a connection with requested pluggable database name is returned.

If the `PDBName` property is set before connection is opened, then the connection's pluggable database name will be the same as `PDBName` when `OracleConnection.Open()` is successfully executed.

If only the `PDBName` property is set, then the default service of the pluggable database is used as the `ServiceName` for this connection.

In unmanaged ODP.NET, if the `PDBName` property is set *and* if the .NET configuration setting, `CPVersion`, is *not* set, then the `CPVersion` attribute will be automatically set to 2.0.

`PDBName` property's value is case insensitive.

The `PDBName` property can be used to retrieve the session's pluggable database name after the connection has been opened. If it is used to get the pluggable database name before the connection is opened, then it just returns the pluggable database name provided by the application, if any.

Only supported for .NET Framework 4 and higher.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

Pipelining

This property enables asynchronous execution on the database server side on a particular `OracleConnection` object.

Declaration

```
// C#  
public bool Pipelining { get; set;}
```

Property Type

System.Boolean

Exceptions

System.ObjectDisposedException - This property cannot be accessed after the OracleConnection object is already disposed.

Remarks

The default value is false.

To enable pipelining, that is, database server asynchronous execution; set this property to true, and enable asynchronous ODP.NET, that is, client side asynchronous execution as well. Pipelining can only be enabled if async ODP.NET is enabled.

The Pipelining property can be configured on OracleConnection and OracleConfiguration objects. By default, OracleConnection Pipelining property value inherits OracleConfiguration Pipelining property value at construction time. The value of OracleConnection Pipelining will be used during the connection.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- [Asynchronous Programming and Pipelining](#)
- [Pipelining](#)

PoolName

This property specifies the name that uniquely identifies the pool in .NET metrics or performance counters.

Declaration

```
// C#  
public string PoolName { get; }
```

Property Type

System.String

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

ProviderName

This property specifies the ODP.NET provider type and version.

Declaration

```
// C#  
public static string ProviderName { get; }
```

Property Type

System.String

Property Values

Managed ODP.NET returns a string consisting of "ODPM.NET : x.x.x.0.0".

ODP.NET Core returns a string consisting of "ODPC.NET : x.x.x.0.0 : y,y".

"x.x.x" is the NuGet package version, such as 23.5.0, of the provider. "y.y" is the target .NET Standard or .NET runtime version, such as 2.1, of the provider.

Remarks

The property is read-only.

The string is returned regardless of the connection state.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

ProviderNuGetVersion

This property specifies the ODP.NET NuGet version number.

Declaration

```
// C#  
public static string ProviderNuGetVersion { get; }
```

Property Type

System.String

Property Values

This property returns a string with the ODP.NET NuGet package version number in the format of x.x.x[-suffix] where each x is a digit in the version. The suffix is optional. A suffix could be -dev, for example, to represent a developer release..

Remarks

The property is read-only.

The string is returned regardless of the connection state.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

ProviderVersion

This property specifies the ODP.NET assembly version.

Declaration

```
// C#  
public static string ProviderVersion { get; }
```

Property Type

System.String

Property Values

The first ODP.NET Core 19c release has a ProviderVersion of 2.0.19.1. Managed and managed ODP.NET have version 4.122.19.1.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

RemoteConfigurationFiltering

This property specifies whether local configuration file access, such as to wallet and tnsnames.ora files, is allowed from centralized configuration providers.

Declaration

```
// C#  
public bool RemoteConfigurationFiltering { get; set; }
```

Property Type

System.Boolean

Remarks

By default, RemoteConfigurationFiltering is set to true. However, the property is ignored unless the connection string is populated with an URL.

When true, local file access is disallowed from centralized configuration providers to wallet, token, tnsnames.ora, and sqlnet.ora files. When false, local file access is allowed.

This property is intended for cloud deployments in which local file access is generally eschewed. Nonetheless, it can be used for on-premises deployments to limit local ODP.NET file access. The fundamental requirement is that an URL must be furnished for each connection string property.

For the centralized configuration providers, RemoteConfigurationFiltering affects the following connection attributes:

- tns_admin
- token_location
- wallet_location

token_location and wallet_location values can be URLs of cloud key vault secrets that store Base64 formatted strings of token data and wallet data, respectively. In a development environment, token_location and wallet_location values can be also the Base64 formatted strings of token data and wallet data. These values are not considered local file access and can be used when RemoteConfigurationFiltering is true.

See Also

- [Centralized Configuration Providers for Deployments](#)
- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

Roles

This property retrieves the OracleRoleCollection object associated with the connection.

Declaration

```
// C#  
public OracleRoleCollection Roles{get;}
```

Property Value

An OracleRoleCollection object containing the role list for the connection.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

The `OracleRoleCollection` object specifies roles to be set on newly created or pool-dispensed connections.

If no `OracleRoleCollection` instance is associated with the connection object, ODP.NET creates a new instance and returns it to the application.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

RowsToFetchPerRoundTrip

This property specifies the total number of rows to retrieve per database round trip.

Declaration

```
// C#  
public Int64 RowsToFetchPerRoundTrip { get; set; }
```

Property Type

`System.Int64`

Remarks

This property has no default value. If it is not set, ODP.NET ignores this property.

The row data fetched in a single round trip applies to scalar types only, such as `NUMBER` and `VARCHAR2` columns. If reference data types, such as LOBs, UDTs, and `XMLType`, exist in the result set, they are retrieved in separate round trips. These reference types can have their own fetch tuning parameters as reference type data sizes can vary in size from row to row, sometimes significantly.

`RowsToFetchPerRoundTrip` can be set before or after the `OracleCommand` executes on `OracleConnection`, `OracleCommand`, `OracleDataReader`, or `OracleRefCursor`. The value can also be changed after initial data fetches so that more or fewer rows are fetched on subsequent round trips. For `OracleConfiguration`, this property can only be set before the first connection opens.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

SDU

This property specifies the session data unit size (SDU) for Oracle networking communication between client and server.

Declaration

```
// C#  
public int SDU { set; }
```

Remarks

Before sending data across the network, ODP.NET or Oracle database buffers and encapsulates data into the SDU. They send the data stored in this buffer when the buffer is full, flushed, or when database server tries to read data. When large amounts of data are being transmitted or when the message size is consistent, adjusting the size of the SDU buffers can improve performance, network utilization, or memory consumption.

The default value is 65536 bytes, equivalent to 64 KB.

ODP.NET and Oracle database can each set its own SDU size. The actual SDU size used is negotiated at connect time. The smaller of the two values is chosen. Oracle AI Database 26ai uses a 64 KB SDU size. Earlier database releases set their default SDU size to 8 KB.

The `OracleConfiguration` SDU size can only be set before the first connection is established globally in the the .NET domain.

The `OracleConnection` SDU size can only be set before the connection is established. Moreover, the `OracleConnection` SDU size for a connection pool can only be set one time. All follow on set attempts are ignored for that pool.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

ServerVersion

This property specifies the version number of the Oracle database to which the `OracleConnection` has established a connection.

Declaration

```
// C#  
public override string ServerVersion {get;}
```

Property Value

The version of the Oracle database.

Exceptions

`InvalidOperationException` - The connection is closed.

Remarks

The default is an empty string.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

ServiceName

This property is used to set and get the database service name to which the session will be connected to when connecting to a pluggable database.

Declaration

```
// C#  
public string ServiceName{ get; set;}
```

Property Value

ServiceName returns a string specifying the service name of the connection.

Exceptions

InvalidOperationException() - the connection is already open.

InvalidOperationException() - if the CPVersion is set to 1.0 and the ServiceName property is set. Applicable only to ODP.NET, Unmanaged Driver.

Remarks

Pluggable databases are available in Oracle Database 12c and higher. The ServiceName and PDBName properties can be used to connect to the specified Service that is running on a particular PDB.

ServiceName must be set before opening the connection so that the correct service is used for the connection.

PDBName property must be set when the ServiceName property is set. Otherwise, an exception is thrown when the connection is opened.

If the ServiceName property is specified before connection is opened, then the connection's service will be the same as the specified ServiceName when the connection is opened.

If the service name is not changed before the connection is next opened, then the connection will continue using the ServiceName value that had been previously set.

In unmanaged ODP.NET, if the ServiceName property is set *and* if the CPVersion attribute is *not* set, then the CPVersion attribute will be automatically set to 2.0.

ServiceName property's value is case insensitive.

The `ServiceName` property can also be used to retrieve the session service name after opening the connection. If retrieved before opening the connection, then `ServiceName` returns the service name provided by the application, if any.

Only supported for .NET Framework 4 and higher.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

ServiceRelocationConnectionTimeout

This property specifies the time, in seconds, for the particular `OracleConnection` to wait before retrying connecting to a service that becomes unavailable.

Declaration

```
// C#  
public string ServiceRelocationConnectionTimeout { get; set;}
```

Property Type

`System.String`

Exceptions

`System.ObjectDisposedException` - This property cannot be accessed after the `OracleConnection` object is already disposed.

`FormatException` - The provided string cannot be converted to a `System.Int32`.

`OverflowException` - The provided string represents a number that is less than `System.Int32.MinValue` or greater than `System.Int32.MaxValue`.

Remarks

Default value is 90 (seconds).

Whenever a database service becomes unavailable, such as due to a service being relocated, an application can encounter numerous connectivity errors during this time. To avoid unnecessary connection attempts to an unavailable service which will result in an error, the driver will block any connection attempts until the service is up or until this property's specified time limit expires from the time when the service DOWN event was received, whichever comes first.

`ServiceRelocationConnectionTimeout` itself can be set based upon the value of `drain_timeout`. For example, if ODP.NET administrators desire the service relocation connection timeout to be `drain_timeout` plus 50 seconds, then they would set `ServiceRelocationConnectionTimeout` in the following manner:

```
//C#  
OracleConnection conn = new OracleConnection();  
conn.ServiceRelocationConnectionTimeout = "drain_timeout + 50";
```

The `ServiceRelocationConnectionTimeout` property can be configured on `OracleConnection` and `OracleConfiguration` objects. By default, `OracleConnection` `ServiceRelocationConnectionTimeout` property value inherits `OracleConfiguration` `ServiceRelocationConnectionTimeout` property value at construction time. The value of `OracleConnection` `ServiceRelocationConnectionTimeout` will be used during the connection.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- [DrainTimeout](#)
- [ServiceRelocationConnectionTimeout](#)

SessionlessTransactionId

This property returns the sessionless transaction identifier.

Declaration

```
// C#  
public byte[] SessionlessTransactionId { get; }
```

Property Value

The byte array which represents the global transaction identifier of the sessionless transaction identifier.

Remarks

If the session not associated with a sessionless transaction, it returns null.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

SessionlessTransactionOptions

This property indicates the sessionless transaction options.

Declaration

```
// C#  
public OracleSessionTransactionOptions SessionlessTransactionOptions { get; set; }
```


Property Value

The `OracleSessionTransactionOption` indicates the options to operate on a sessionless transaction.

Remarks

Default value is null.

This property auto-initializes a sessionless transaction if its properties are not set.

The `SessionTransactionOptions` property is only read and used when `BeginSessionlessTransaction()` or `ResumeSessionlessTransaction` is invoked. Any change in `SessionlessTransactionOptions` property values afterwards will not be reflected in the sessionless transaction.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

SqlNetAllowedLogonVersionClient

This property is used to specify the minimum authentication protocol that is to be used for a given `OracleConnection` object.

Declaration

```
// C#  
public OracleAllowedLogonVersionClient SqlNetAllowedLogonVersionClient { set; }
```

Property Type

`OracleAllowedLogonVersionClient`

Property Value

An `OracleAllowedLogonVersionClient` enumeration value.

Example

```
// C#  
OracleConfiguration.SqlNetAllowedLogonVersionClient =  
OracleAllowedLogonVersionClient.Version12a;
```

Remarks

The default is `OracleAllowedLongVersionClient.12`.

① See Also

- Oracle Database Net Services Reference to better understand the minimum authentication protocol that is to be used when authenticating against the Oracle database for a given OracleConnection object
- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- [OracleAllowedLogonVersionClient Enumeration](#)

SqlNetAuthenticationServices

This property enables one or more authentication services, such as TCP/IP with SSL.

Declaration

```
// C#  
public string SqlNetAuthenticationServices { get; set; }
```

Property Type

System.String

Valid values are: All, Kerberos5, NTS, TCPS, or NONE.

① Note

ODP.NET Core does not currently support the Kerberos5 value.

Example

```
//C#  
con.SqlNetAuthenticationServices = "(nts, tcps)";
```

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleConnection Class](#)
- [OracleConnection Members](#)

SqlNetCryptoChecksumClient

This property specifies the checksum client behavior.

Declaration

```
// C#  
public string SqlNetCryptoChecksumClient {get; set;}
```

Return Value

System.String

Remarks

Default value is null. Null setting means other configuration methods determine the client's data integrity behavior.

Possible string values:

- accepted - enable the security service if required or requested by the database.
- rejected - disable the security service, even if required by the database.
- requested - enable the security service if the database allows it.
- required - enable the security service and disallow the connection if the database is not enabled for the security service.

All other string values are incorrect and should not be used.

Example

```
conn.SqlNetCryptoChecksumClient = "REQUESTED";
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

SqlNetCryptoChecksumTypesClient

This property specifies the checksum algorithms the client can use.

Declaration

```
// C#  
public string SqlNetEncryptionTypesClient {get; set;}
```

Return Value

System.String

Remarks

Default value is null. Null setting means other configuration methods determine the client's data integrity algorithms.

Possible string values:

- SHA1
- SHA256
- SHA384
- SHA512

All other string values are incorrect and should not be used.

Example

```
// Allow ODP.NET app to use SHA256 and SHA512 checksum algorithms
conn.SqlNetEncryptionClient = "(SHA256, SHA512)";
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

SqlNetEncryptionClient

This property specifies the encryption client behavior.

Declaration

```
// C#
public string SqlNetEncryptionClient {get; set;}
```

Return Value

System.String

Remarks

Default value is null. Null setting means other configuration methods determine the client's encryption behavior.

Possible string values:

- `accepted` - enable the security service if required or requested by the database.
- `rejected` - disable the security service, even if required by the database.
- `requested` - enable the security service if the database allows it.
- `required` - enable the security service and disallow the connection if the database is not enabled for the security service.

All other string values are incorrect and should not be used.

Example

```
conn.SqlNetEncryptionClient = "REQUESTED";
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

SqlNetEncryptionTypesClient

This property specifies encryption algorithms that the client can use.

Declaration

```
// C#  
public string SqlNetEncryptionTypesClient {get; set;}
```

Return Value

System.String

Remarks

Default value is null. Null setting means other configuration methods determine the client's encryption algorithms.

Possible string values:

- AES128
- AES192
- AES256
- 3DES112
- 3DES168

All other string values are incorrect and should not be used.

Using 3DES 112-bit (3DES112) or 3DES 168-bit (3DES168) requires ODP.NET SQLNET.Allow_Weak_Crypto/SqlNetAllowWeakCrypto to be set to true.

Example

```
// Allow ODP.NET app to use AES128 and AES192 encryption algorithms  
conn.SqlNetEncryptionTypesClient = "(AES128, AES192)";
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- OracleConfiguration [SqlNetAllowWeakCrypto](#)

SqlNetWalletOverride

This property specifies whether the client overrides the strong authentication credential with the stored wallet password credential for the database connection.

Declaration

```
// C#  
public bool SqlNetWalletOverride { get; set; }
```

Property Type

System.Boolean

Remarks

Default is false. By setting this value to true, ODP.NET uses the stored wallet password credential to connect.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

SSLCertificateThumbprint

This property specifies the TLS/SSL certificate thumbprint in the certificate store for the ODP.NET connection to use.

Declaration

```
// C#  
public string SSLCertificateThumbprint { get; set; }
```

Remarks

When there are multiple certificates possible for ODP.NET to use from the certificate store, specifying the thumbprint tells ODP.NET which certificate to connect the user with. This avoids the user from performing certificate selection manually, such as via a graphical interface. Users can select a certificate manually using the `OracleConnection.AllowCertificateSelectionUI` property.

ODP.NET supports certificates through thumbprint only. It supports the SHA1 and SHA2 algorithms.

ODP.NET expects thumbprints in the format of `<Algorithm>:<Hash>`. For example,

- SHA1:1B:11:01:5E:D1:7C:20:B2:62:39:3E:04:7B:83:47:DE:70:2E:4E:44
- SHA256:CD:C6:8B:0D:36:FE:20:0E:7F:86:5F:94:0B:4D:EC:EE:1B:C0:D1:09

Or

- SHA1:1B11015ED17C20B262393E047B8347DE702E4E44
- SHA256:CDC68B0D36FE200E7F865F940B4DECEE1BC0D109

Code Sample

```
OracleConnection myCon = new OracleConnection();  
myCon.SSLCertificateThumbprint =  
"SHA1:1B:11:01:5E:D1:7C:20:B2:62:39:3E:04:7B:83:47:DE:70:2E:4E:44";
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

SSLServerCertDN

This property sets the distinguished name (DN) string that will be used to validate the server certificate DN.

Declaration

```
// C#  
public string SSLServerCertDN {get; set;}
```

Property Type

System.String

Remarks

This property overrides the SSL_SERVER_CERT_DN parameter, including the full connect descriptor and Easy Connect Plus configuration.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

SSLServerDNMatch

This property enables server certificate distinguished name (DN) validation.

Declaration

```
// C#  
public bool SSLServerDNMatch {get; set;}
```

Property Type

System.Boolean

Remarks

This property enables server certificate validation through DN matching. By default, this property is null to allow other configuration parameters to enforce DN matching.

Set to true to enforce a match. To enforce DN matching, the SSLServerCertDN property value must match the DN from the server certificate for the connection to succeed. Otherwise, the connection fails.

Set to false to not enforce a match.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

SSLVersion

This property enforces the connection to use a specific SSL/TLS version.

Declaration

```
// C#  
public string SSLVersion {get; set;}
```

Property Type

System.String

Description

Supported values are: undetermined |1.2|TLSv1.2|1.3|TLSv1.3. To specify more than one SSL/TLS version, enclose the version numbers between parentheses and separate the values with a comma. For example, (1.2,1.3).

Starting with ODP.NET 23.9, you can specify a minimum TLS version by appending the plus (+) sign after the version, such as 1.2+ or TLSv1.2+. This value indicates to use TLS 1.2 or higher, whichever highest version the database server supports.

In some cases, apps may want to set a minimum TLS version, but not support one or more higher versions. To identify excluded versions, set the `TLSExcludeVersion` property on `OracleConfiguration`, `OracleConnection`, or equivalent.

The default value is an undetermined version. An undetermined version means ODP.NET is allowed to use any TLS version, this property is allowed to specify. An exception will be thrown for invalid value.

This property overrides the `SSL_VERSION` parameter, including the full connect descriptor and Easy Connect Plus configuration.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

State

This property specifies the current state of the connection.

Declaration

```
// C#  
public override ConnectionState State {get;}
```

Property Value

The ConnectionState of the connection.

Implements

IDbConnection

Remarks

ODP.NET supports ConnectionState.Closed and ConnectionState.Open for this property. The default value is ConnectionState.Closed.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

StatementCacheSize

This property specifies the current size of the statement cache associated with this connection.

Declaration

```
// C#  
public int StatementCacheSize{get;}
```

Property Value

An integer value indicating the size of the statement cache.

Remarks

If self tuning is not enabled, then the default value of this property depends upon the statement cache size specified in the connection string, application configuration file, or the registry. If none of these values are specified, then a default value of 0 is used.

If self tuning is enabled, then the property value is adjusted automatically. Any values specified in the connection string, application configuration file, or the registry are ignored.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

SuppressGetDecimalInvalidCastException

This property specifies whether to suppress the `InvalidCastException` and return a rounded-off 28 or 29 precision Oracle `NUMBER` value that can be represented as a .NET `Decimal`.

Declaration

```
// C#  
public bool SuppressGetDecimalInvalidCastException { get; set; }
```

Property Type

`System.Boolean`

Remarks

Oracle `NUMBER` has a maximum of 38 precision. .NET `Decimal` has a maximum of 28 or 29 precision. When the `OracleDataReader.GetDecimal()`, `GetValue()`, or `OracleDataAdapter.Fill()` method is called for an Oracle `NUMBER` value that cannot be represented as a .NET `Decimal`, then ODP.NET throws an `InvalidCastException` because not all the precision can be preserved when converting the number to a .NET `Decimal`.

This behavior occurs when `SuppressGetDecimalInvalidCastException` is set to `false`, which is the default value.

When `SuppressGetDecimalInvalidCastException` is set to `true`, the resulting .NET `Decimal` is rounded off to 28 or 29 precision, allowing as much of the value to be represented without throwing an exception. If the resulting rounded number is larger than can be stored in a .NET `Decimal`, an exception will be thrown, such as the number 1×10^{32} .

The `SuppressGetDecimalInvalidCastException` property can also be configured on `OracleConfiguration`, `OracleDataReader`, and `OracleDataAdapter` objects. The `OracleConnection.SuppressGetDecimalInvalidCastException` property value inherits `OracleConfiguration.SuppressGetDecimalInvalidCastException` property value at construction time.

By using `OracleConnection.SuppressGetDecimalInvalidCastException`, this ODP.NET setting becomes enabled for all the connection's command executions. It can be overridden at the `OracleDataReader` or `OracleDataAdapter` level, which each has its own `SuppressGetDecimalInvalidCastException` property.

① See Also

- OracleConfiguration [SuppressGetDecimalInvalidCastException](#)
- OracleDataReader [SuppressGetDecimalInvalidCastException](#)
- OracleDataAdapter [SuppressGetDecimalInvalidCastException](#)
- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleConnection Class](#)
- [OracleConnection Members](#)

SwitchedConnection

This property applies to pluggable database connections. Upon an `OracleConnection.Open()`, if the connection returned from the pool had to change its service name (`ServiceName`) or pluggable database (`PDBName`) in order to connect to the desired pluggable database, then the connection is considered switched. Thus, `SwitchedConnection` will then return `TRUE`. In all other cases, it will return `FALSE`.

Declaration

```
// C#  
public bool SwitchedConnection { get; }
```

Remarks

Applications should use this property carefully when using `DRCPConnectionString` values that are common across PDBs/Services. If the same `DRCPConnectionString` values are used across PDBs/Services, but their semantics are different, then this property value can help determine if the requested connection is returned or not.

For example, in a scenario where the same `DRCPConnectionString` that is used across PDBs/Services has a different semantic, the application should first check to see if the session has been switched or not. If it has switched between PDBs/Services, then the application should consider that the connection with a matching `DRCPConnectionString` was not dispensed.

Only supported for .NET Framework 4 and higher.

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleConnection Class](#)
- [OracleConnection Members](#)

TAFMode

This property configures Transparent Application Failover behavior on the connection.

Declaration

```
// C#  
public OracleTAFMode TAFMode
```

Remarks

Applications set this property on the connection to configure TAF behavior. The set value is only considered for the first pooled connection, which is then cached for the pool. Subsequent connections ignore the set value and instead use the cached value.

To use a new value, applications must call `ClearPool()` or `ClearAllPools()` to remove the cached `OracleTAFMode` value from the pool. The reset pool can then use the newly provided value.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

TLSExcludeVersion

This property indicates which SSL/TLS versions connections should not be enabled for.

Declaration

```
// C#  
public string TLSExcludeVersion { get; set; }
```

Property Type

System.String

Description

The `TLSVersion` or `SSLVersion` properties sets the supported connection SSL/TLS versions. `TLSExcludeVersion` sets the versions that the app connections cannot use.

Typically, `TLSExcludeVersion` is used when the TLS/SSL version has a minimum value set, such as `TLSv1.2+`, and the app wants to prevent connections from using a version above the minimum value, such as `TLS 1.3`.

Supported exclusion values are: `1.2|TLSv1.2|1.3|TLSv1.3`. To specify more than one SSL/TLS version to exclude, enclose the version numbers between parentheses and separate the values with a comma. For example, `(1.2,1.3)`.

This property has no default value.

An exception will be thrown for invalid value.

This property overrides the `TLS_EXCLUDE_VERSION` parameter, including the full connect descriptor and Easy Connect Plus configuration.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

TLSVersion

This property enforces the connection to use a specific SSL/TLS version.

Declaration

```
// C#  
public static string TLSVersion { get; set; }
```

Property Type

System.String

Description

Supported values are: undetermined |1.2|TLSv1.2|1.3|TLSv1.3. To specify more than one SSL/TLS version, enclose the version numbers between parentheses and separate the values with a comma. For example, (1.2,1.3).

Starting with ODP.NET 23.9, you can specify a minimum TLS version by appending the plus (+) sign after the version, such as 1.2+ or TLSv1.2+. This value indicates to use TLS 1.2 or higher, whichever highest version the database server supports.

In some cases, apps may want to set a minimum TLS version, but not support one or more higher versions. To identify excluded versions, set the TLSExcludeVersion property on OracleConfiguration, OracleConnection, or equivalent.

The default value is an undetermined version. An undetermined version means ODP.NET is allowed to use any TLS version, this property is allowed to specify. An exception will be thrown for invalid value.

This property overrides the SSL_VERSION parameter, including the full connect descriptor and Easy Connect Plus configuration.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

TnsAdmin

This property specifies the directory location of tnsnames.ora or sqlnet.ora, or both.

Declaration

```
// C#  
public static string TnsAdmin { get; set; }
```

Property Type

System.String

Remarks

The backslash (\) is a special character in .NET that represents the beginning of an escape sequence. To specify a directory location, use any one of the following formats in .NET so that backslashes are correctly represented in a directory location:

```
// C#  
OracleConnection con = new OracleConnection();  
con.TnsAdmin = "D:\\oracle\\client\\admin";  
con.TnsAdmin = @"D:\oracle\client\admin";  
con.TnsAdmin = "D:/oracle/client/admin";
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

TokenAuthentication

This property sets the value for TOKEN_AUTH parameter for the connection.

Declaration

```
// C#  
public OracleTokenAuth TokenAuthentication { get; set; }
```

Remarks

The value of this property will be part of connection pool manager identification so different values for this property will result in different connection pools.

Table 6-52 OracleTokenAuth Members

Member Name	Description
AzureServicePrincipal	Token authentication enabled for Microsoft Entra ID service principal <i>Not available in ODP.NET, Unmanaged Driver</i>
AzureManagedIdentity	Token authentication enabled for Microsoft Entra ID managed identity <i>Not available in ODP.NET, Unmanaged Driver</i>
AzureInteractive	Token authentication enabled for Microsoft Entra ID using interactive authentication method <i>Not available in ODP.NET, Unmanaged Driver</i>

Table 6-52 (Cont.) OracleTokenAuth Members

Member Name	Description
AzureDeviceCode	Token authentication enabled for Microsoft Entra ID using device code method <i>Not available in ODP.NET, Unmanaged Driver</i>
AzureDefault	Token authentication enabled for Microsoft Entra ID using default method <i>Not available in ODP.NET, Unmanaged Driver</i>
Disabled	Default value. Token authentication is DISABLED.
OciApiKey	Token authentication using OCI_API_KEY authentication flow. <i>Not available in ODP.NET, Unmanaged Driver</i>
OciInstancePrincipal	Token authentication using OCI_INSTANCE_PRINCIPAL authentication flow. <i>Not available in ODP.NET, Unmanaged Driver</i>
OciResourcePrincipal	Token authentication using OCI_RESOURCE_PRINCIPAL authentication flow. <i>Not available in ODP.NET, Unmanaged Driver</i>
OciDelegationToken	Token authentication using OCI_DELEGATION_TOKEN authentication flow. <i>Not available in ODP.NET, Unmanaged Driver</i>
OciInteractive	Token authentication using OCI_INTERACTIVE authentication flow. <i>Not available in ODP.NET, Unmanaged Driver</i>
OciDefault	Token authentication using the most appropriate method depending on the application environment. ODP.NET first attempts to retrieve the token using the OCI API key. If that fails, then ODP.NET checks whether it is in a Cloud Shell environment to attempt to get the token using delegation token authentication flow. Lastly, it attempts to use the instance principal authentication flow. <i>Not available in ODP.NET, Unmanaged Driver</i>
OciToken	Token authentication is enabled for Oracle Identity and Access Management.
OAuth	Token authentication enabled for Microsoft Entra ID.

Exceptions

An `InvalidOperationException` will be raised if

- the connection is already open,
- the value of the `AccessToken` property is not compatible with the type of token authentication being set,
- `TokenAuthentication` is set to disabled and the `AccessToken` or `TokenLocation` properties are set,
- `TokenAuthentication` is set to a value other than disabled, and the `Credential` property is set,
- the user or proxy information in the connection string is not compatible with token authentication,
- if user id value is other than /,
- password, proxy user id, and proxy password is set,

- TokenConfiguration is specified, and TokenAuthentication is set to OciToken, OAuth, or Disabled.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

TokenLocation

This property gets or sets the token location path for the connection where the access token, or database token and private key files, are present for ODP.NET token authentication. The file names do not need to be specified.

Not Available in ODP.NET, Unmanaged Driver

Declaration

```
// C#  
public string TokenLocation { get; set; }
```

Property Value

A directory location.

Remarks

The value of this property will be part of connection pool manager identification so different values for this property will result in different connection pools.

Expections

An `InvalidOperationException` will be raised if

- the connection is already open,
- the `AccessToken` or `Credential` properties are set,
- `TokenAuthentication` has been set to disabled by the application,
- the user or proxy information in the connection string is not compatible with token authentication,
- user id value is other than /, or
- password, proxy user id, and proxy password is set.

Description

This property is mandatory for OAuth file-based access tokens, but optional for database tokens.

In the access token case, this property can be the directory where the file token is, or the full-path specification of the token file.

For OCI IAM PoP tokens, a file's full-path specification can be specified if it does not have the standard name, token. A file name does not need to be specified when it uses this standard, fixed name. Bearer token default token location is an empty string.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

UseClientInitiatedCQN

This property specifies, on a particular `OracleConnection` object, whether to use Client Initiated Continuous Query Notification (CICQN) or traditional Continuous Query Notification (CQN) that does not rely on a persistent connection.

Declaration

```
// C#  
public bool UseClientInitiatedCQN { get; set; }
```

Property Type

`System.Boolean`

Exceptions

`System.ObjectDisposedException` - This property cannot be accessed after the `OracleConnection` object is already disposed.

Remarks

The default value is `false`.

When set to `true`, ODP.NET creates and uses a separate connection to receive server-initiated database change notifications. This is known as CICQN. This single connection aggregates all notifications for the pool, but the connection does not count toward the Min Pool Size nor the Max Pool Size limits. For managed ODP.NET, if there are five connection pools using CICQN, then ODP.NET will then have five additional connections, one per pool.

When set to `false`, then ODP.NET will use traditional CQN to receive the database change notifications. The client creates a listening end point (i.e., IP address and port) that does not rely on a database connection to continuously exist.

CICQN is commonly used in cloud scenarios or if firewalls between the database and client do not allow access to the client port. Otherwise, traditional CQN is used.

ODP.NET CICQN requires Oracle Database 21c or higher.

`UseClientInitiatedCQN` must be set to its intended value prior to opening a database connection.

The `UseClientInitiatedCQN` property can be configured on `OracleConnection` and `OracleConfiguration` objects. By default, `OracleConnection UseClientInitiatedCQN` property value inherits `OracleConfiguration UseClientInitiatedCQN` property value at

construction time. The value of `OracleConnection.UseClientInitiatedCQN` will be used during the connection.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

UseHourOffsetForUnsupportedTimezone

This property specifies whether the hour offset can be used for the session time zone, when the Oracle time zone region name that is associated with the .NET locale is not supported by the Oracle Database being used.

Declaration

```
// C#  
public bool UseHourOffsetForUnsupportedTimezone { get; set; }
```

Property Type

`System.Boolean`

Exceptions

`InvalidOperationException` – This exception will be thrown if this property is set when the Connection is in an Open state.

Remarks

ODP.NET is built to support the latest available time zones the Oracle database client supports. However, older Oracle database server versions may not have the latest time zone files that support all the same time zones the client supports. Thus, the client could request a connection time zone the Oracle database does not support. In these situations, the connection request returns an "ORA-01882: timezone region not found" error.

When the `UseHourOffsetForUnsupportedTimezone` property is set to true, then ODP.NET will initialize the connection/session time zone using a "hour offset" instead of a time zone region name only if the Oracle Database cannot support the Oracle time zone associated with the .NET locale. ODP.NET can then successfully connect and avoid the ORA-01882 error. This also means if the Oracle Database does support the Oracle time zone associated with the .NET locale, then the connection/session will be initialized with an Oracle time zone region name, regardless of the `UseHourOffsetForUnsupportedTimezone` property setting.

By default, the `UseHourOffsetForUnsupportedTimezone` property is set to false.

Oracle does not recommend using the hour offset since datetimes with time zone values will *not* be daylight savings time sensitive. Even when the hour offset is enabled, ODP.NET first attempts to use the Oracle time zone region name associated with the .NET locale. Only when the ORA-01882 error is encountered will ODP.NET use the hour offset as the session time zone during a second connection attempt. Thus, there can be two connection requests if the first connection attempt fails with the ORA-01882 error.

The `UseHourOffsetForUnsupportedTimezone` property value specified for the first successfully created connection is used for all subsequent connections in that pool. When pooling is not used, the property's value will be specific for the particular `OracleConnection.Open()` invocation.

Oracle recommends resolving ORA-01882 errors by upgrading the Oracle database with the latest time zone/DST files. Oracle also recommends using the time zone region name for the session time zone so that datetime conversions with time zone values are daylight savings time sensitive.

If the Oracle Database time zone / DST files cannot be upgraded, the application should then set the `UseHourOffsetForUnsupportedTimezone` property to `true`.

Sample Code

```
using System;
using Oracle.ManagedDataAccess.Client;

class Test
{
    static void Main(string[] args)
    {
        string constr = "user id=hr;password=<password>;data source=oracle;"

        OracleConnection con = new OracleConnection(constr);

        // Have the connection created with the 'hour offset' being used as the session time
        // zone
        // if the Oracle Database does not support the Oracle Time Zone associated with the
        // .NET application's locale
        con.UseHourOffsetForUnsupportedTimezone = true;

        // Establish a connection to Oracle
        con.Open();
        con.Dispose();
    }
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

UseSNI

This property specifies whether to use Server Name Indication (SNI).

Declaration

```
// C#
bool? UseSNI { get; set; }
```

Property Type

`System.Boolean`

Remarks

Default is false. False disables using SNI.

When UseSNI is set to true, the following configuration rules must be obeyed. Any deviation will disable SNI automatically.

- Only the following CONNECT_DATA names can be used.
 - ServiceName
 - Server
 - InstanceName
 - ColocationTag
- The CONNECT_DATA name values can only contain alphanumeric, underscore, hyphen, or period characters.
- No more than two consecutive period characters
- No hyphen character before or after a period
- No hyphen at the beginning or end of the CONNECT_DATA name values above
- No period at the beginning or end of the CONNECT_DATA name values above

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- USE_SNI parameter in sqlnet.ora

WalletLocation

This property specifies the location of wallets. Wallets are certificates, keys, and trustpoints processed by SSL/TLS.

Declaration

```
// C#  
public static string WalletLocation { get; set; }
```

Property Type

System.String

Remarks

Microsoft Certificate Store (MCS) and file system wallets are supported.

The backslash (\) is a special character in .NET that represents the beginning of an escape sequence. To specify a directory location, use any one of the following formats in .NET so that backslashes are correctly represented in a directory location:

```
// C#
OracleConnection con = new OracleConnection();
con.WalletLocation = "D:\\oracle\\client\\wallets";
con.WalletLocation = @"D:\oracle\client\wallets";
con.WalletLocation = "D:/oracle/client/wallets";
```

This property can accept file path or URL values that contain the wallet file.

It supports single sign-on files (.sso file extension) and PKCS #12 files (.p12 file extension). Since wallet location is a directory location, not file location, ODP.NET chooses the file to use based on the criteria:

- When the wallet password is not supplied:
 - Reads the .sso file, if present
 - Reads the .p12 file, if present
- When the wallet password is supplied:
 - Reads the .p12 file, if present

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

WalletPassword

This property stores the password to decrypt the PKCS #12 wallet.

Declaration

```
// C#
public OracleOpaqueString WalletPassword {get; set;}
```

Property Value

The password to decrypt the encrypted PKCS #12 wallet.

Remarks

Wallet passwords support the same functionality as traditional ODP.NET passwords.

Since PKCS #12 files are encrypted, apps must provide the wallet password as an `OracleOpaqueString`. When the wallet password is provided via the `OracleConfiguration.WalletPassword` or `OracleConnection.WalletPassword` class property, ODP.NET loads the associated PKCS #12 file as designated by the wallet location. Alternately, if no password is provided, ODP.NET loads the associated SSO file.

Changing the `WalletPassword` for an existing connection pool does not create a new pool. Only setting the property or unsetting it creates a new pool.

Note

The `OracleConfiguration.WalletPassword` is associated with the `OracleConfiguration.WalletLocation` and the `OracleConnection.WalletPassword` is associated with the `OracleConnection.WalletLocation`. For example, if `OracleConfiguration.WalletPassword` is set, and `OracleConnection.WalletLocation` is set, then the wallet password from `OracleConfiguration` will not be used when opening the wallet specified on the `OracleConnection`.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

OracleConnection Public Methods

`OracleConnection` public methods are listed in [Table 6-53](#).

Table 6-53 OracleConnection Public Methods

Public Method	Description
BeginRequest	Starts an explicit request boundary when using Application Continuity <i>Not available in ODP.NET, Unmanaged Driver</i>
BeginSessionlessTransaction	Starts a new sessionless transaction and returns its global transaction identifier <i>Not available in ODP.NET, Unmanaged Driver</i>
BeginTransaction	Begins a local transaction (Overloaded)
ChangeDatabase	<i>Not Supported</i>
Clone	Creates a copy of an <code>OracleConnection</code> object
Close	Closes the database connection
CreateCommand	Creates and returns an <code>OracleCommand</code> object associated with the <code>OracleConnection</code> object
<code>CreateObjRef</code>	Inherited from <code>System.MarshalByRefObject</code>
Commit()	Commits the database transaction
DisableReplay	Disables replay on the connection in Application Continuity <i>Not available in ODP.NET, Unmanaged Driver</i>
<code>Dispose</code>	Inherited from <code>System.ComponentModel.Component</code>
EndRequest	Indicates an ending explicit request boundary when using Application Continuity <i>Not available in ODP.NET, Unmanaged Driver</i>

Table 6-53 (Cont.) OracleConnection Public Methods

Public Method	Description
EnlistDistributedTransaction	Enables applications to explicitly enlist in a specified distributed transaction <i>Not supported in ODP.NET Core</i>
EnlistTransaction	Enables applications to enlist in a specified distributed transaction
Equals	Inherited from System.Object (Overloaded)
FlushCache	Flushes all updates and deletes made through REF objects retrieved using this connection <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
GetHashCode	Inherited from System.Object
GetLifetimeService	Inherited from System.MarshalByRefObject
GetSchema	Returns schema information for the data source of the OracleConnection
GetSessionInfo	Returns or refreshes the property values of the OracleGlobalization object that represents the globalization settings of the session (Overloaded)
GetType	Inherited from System.Object
InitializeLifetimeService	Inherited from System.MarshalByRefObject
Open	Opens a database connection with the property settings specified by the ConnectionString
OpenAsync	Returns a Task-based asynchronous version of OracleConnection.Open (Overloaded)
OpenWithNewClientSecret	Opens a new connection using a new Microsoft Entra ID client secret <i>Not available in ODP.NET, Unmanaged Driver</i>
OpenWithNewPassword	Opens a new connection with the new password (Overloaded)
OpenWithNewPasswordAsync	A Task-based asynchronous version of OracleConnection.OpenWithNewPassword(), which opens a new connection with the new password. (Overloaded)
OpenWithNewToken	This method can be used to open a new connection by providing the refreshed/updated access token directly to ODP.NET (Overloaded)
PurgeStatementCache	Flushes the Statement Cache by closing all open cursors on the database, when statement caching is enabled
ResumeSessionlessTransaction	Resumes a sessionless transaction using the provided unique transaction identifier <i>Not Available in ODP.NET, Unmanaged Driver</i>
Rollback	Rolls back the local transaction
Save(String)	Creates a savepoint in the database
SetSessionInfo	Alters the session's globalization settings with the property values provided by the OracleGlobalization object

Table 6-53 (Cont.) OracleConnection Public Methods

Public Method	Description
SetShardingKey(OracleShardingKey, OracleShardingKey)	Enables applications to set the sharding key and super sharding key before requesting a connection
SuspendSessionlessTransaction	Suspends the active sessionless transaction immediately from the session. <i>Not available in ODP.NET, Unmanaged Driver</i>
ToString	Inherited from System.Object
UseAzureTokenAuthentication	Associates the Microsoft Entra ID token authentication settings with the OracleConnection object for use upon the next connection open <i>Not available in ODP.NET, Unmanaged Driver</i>
UseOciTokenAuthentication	Associates the OCI IAM token authentication settings with the OracleConnection object for use upon the next connection open. <i>Not available in ODP.NET, Unmanaged Driver</i>

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

BeginRequest

This method starts an explicit request boundary when using Application Continuity.

Declaration

```
// C#
public void BeginRequest();
```

Remarks

When called after another BeginRequest, but before an EndRequest, this call is a no-op and does not throw any exception. Therefore, applications are allowed to call BeginRequest after a connection pool checkout, which implicitly calls BeginRequest.

See Also

- [Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

BeginSessionlessTransaction

This method starts a new sessionless transaction and returns its global transaction identifier.

Declaration

```
// C#  
public byte[] BeginSessionlessTransaction();
```

Return Value

A unique transaction identifier of the sessionless transaction.

Exceptions

InvalidOperationException - The connection is already associated with a transaction or is currently in a transaction.

OracleException - The database does not support sessionless transactions (ORA-26207).

Remarks

The unique transaction identifier is used to resume the transaction using the `OracleConnection.ResumeSessionlessTransaction()` method.

The sessionless transaction isolation level is based on the session's current isolation level. `OracleConnection` can operate on a sessionless transaction through its `Commit()`, `Rollback()`, `SuspendSessionlessTransaction()`, and `ResumeSessionlessTransaction()` methods.

Auto-commit is disabled when this method is invoked successfully.

If the `OracleConnection.SessionlessTransactionOptions` is set, then `BeginSessionlessTransaction()` operates the sessionless transaction based upon the `OracleConnection.SessionlessTransactionOptions` specified settings.

If the `OracleConnection.SessionlessTransactionOptions` is not set, then `BeginSessionlessTransaction()` operates the sessionless transaction using the following default behavior:

- Sessionless transaction timeout is 60 seconds.
- The connection is associated with the `OracleConnection.BeginSessionlessTransaction()` sessionless transaction. The transaction starts upon the first `OracleCommand` SQL execution.
- The sessionless transaction will not be suspended implicitly after SQL execution.

① See Also

- [Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

BeginTransaction

`BeginTransaction` methods begin local transactions.

Overload List

- [BeginTransaction\(\)](#)
This method begins a local transaction.
- [BeginTransaction\(IsolationLevel\)](#)
This method begins a local transaction with the specified isolation level.

BeginTransaction()

This method begins a local transaction.

Declaration

```
// C#  
public OracleTransaction BeginTransaction();
```

Return Value

An `OracleTransaction` object representing the new transaction.

Implements

`IDbConnection`

Exceptions

`InvalidOperationException` - A transaction has already been started.

Remarks

The transaction is created with its isolation level set to its default value of `IsolationLevel.ReadCommitted`. All further operations related to the transaction must be performed on the returned `OracleTransaction` object.

If a local transaction is already started implicitly, invoking `BeginTransaction()` will inherit that transaction.

If the transaction is created explicitly using `BeginTransaction()`, the transaction can be operated on either through the `OracleConnection` methods or `OracleTransaction` methods. But once the transaction is over via a `Commit()` or a `Rollback()` invoked on either object, the `OracleTransaction` cannot be used henceforth.

The `BeginTransaction(IsolationLevel)` method overrides the `OracleConnection` `IsolationLevel` property. Once that transaction commits or rolls back, the connection's `IsolationLevel` returns to its previous value.

Auto-commit is disabled when this method is invoked successfully.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

BeginTransaction(IsolationLevel)

This method begins a local transaction with the specified isolation level.

Declaration

```
// C#  
public OracleTransaction BeginTransaction(IsolationLevel isolationLevel);
```

Parameters

- *isolationLevel*
The isolation level for the new transaction.

Return Value

An `OracleTransaction` object representing the new transaction.

Implements

`IDbConnection`

Exceptions

`InvalidOperationException` - A transaction has already been started.

`ArgumentException` - The `isolationLevel` specified is invalid.

Remarks

The following isolation levels are supported: `IsolationLevel.ReadCommitted` and `IsolationLevel.Serializable`.

Although the `BeginTransaction` method supports the `IsolationLevel.Serializable` isolation level, serializable transactions are not supported when using `System.Transactions` and `TransactionScope`.

Requesting other isolation levels causes an exception.

If a local transaction is already started implicitly, invoking `BeginTransaction()` will inherit that transaction.

If the transaction is created explicitly using `BeginTransaction()`, the transaction can be operated on either through the `OracleConnection` methods or `OracleTransaction` methods. But once the transaction is over via a `Commit()` or a `Rollback()` invoked on either object, the `OracleTransaction` cannot be used henceforth.

The `BeginTransaction(IsolationLevel)` method overrides the `OracleConnection` `IsolationLevel` property. Once that transaction commits or rolls back, the connection's `IsolationLevel` returns to its previous value.

Auto-commit is disabled when this method is invoked successfully.

Example

```
// C#  
  
using System;  
using System.Data;
```

```
using Oracle.DataAccess.Client;

class BeginTransactionSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        // Create an OracleCommand object using the connection object
        OracleCommand cmd = con.CreateCommand();

        // Start a transaction
        OracleTransaction txn = con.BeginTransaction(IsolationLevel.ReadCommitted);

        // Update EMP table
        cmd.CommandText = "update emp set sal = sal + 100";
        cmd.ExecuteNonQuery();

        // Rollback transaction
        txn.Rollback();
        Console.WriteLine("Transaction rolledback");

        // Clean up
        txn.Dispose();
        cmd.Dispose();
        con.Dispose();
    }
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

ChangeDatabase

This method is not supported.

Declaration

```
// C#
public override void ChangeDatabase(string databaseName);
```

Parameters

- *databaseName*
The name of the database that replaces the current database name.

Implements

IDbConnection.ChangeDatabase

Exceptions

NotSupportedException - Method not supported.

Remarks

This method is not supported and throws a NotSupportedException if invoked.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

Clone

This method creates a copy of an OracleConnection object.

Declaration

```
// C#  
public object Clone();
```

Return Value

An OracleConnection object.

Implements

ICloneable

Remarks

The cloned object has the same property values as that of the object being cloned.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Client;  
  
class CloneSample  
{  
    static void Main()  
    {  
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";  
        OracleConnection con = new OracleConnection(constr);  
        con.Open();  
  
        // Need a proper casting for the return value when cloned  
        OracleConnection clonedCon = (OracleConnection)con.Clone();  
  
        // Cloned connection is always closed, regardless of its source,  
        // But the connection string should be identical
```

```
        clonedCon.Open();
        if (clonedCon.ConnectionString.Equals(con.ConnectionString))
            Console.WriteLine("The connection strings are the same.");
        else
            Console.WriteLine("The connection strings are different.");

        // Close and Dispose OracleConnection object
        clonedCon.Dispose();
    }
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

Close

This method closes the connection to the database.

Declaration

```
// C#
public override void Close();
```

Implements

IDbConnection

Remarks

Performs the following:

- Rolls back any pending local or active sessionless transactions that are not yet committed. The connection will no longer associate with any sessionless transaction or local transaction. Distributed transactions will rely on the distributed transaction coordinator on whether roll back is necessary.
- Places the connection in the pool if connection pooling is enabled. Even if connection pooling is enabled, the connection can be closed if it exceeds the connection lifetime specified in the connection string. If connection pooling is disabled, the connection is closed.
- Closes the connection to the database.

The connection can be reopened using `Open()`.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

CreateCommand

This method creates and returns an `OracleCommand` object associated with the `OracleConnection` object.

Declaration

```
// C#  
public OracleCommand CreateCommand();
```

Return Value

The `OracleCommand` object.

Implements

`IDbConnection`

Example

```
// C#  
  
using System;  
using System.Data;  
using Oracle.DataAccess.Client;  
  
class CreateCommandSample  
{  
    static void Main()  
    {  
        // Connect  
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";  
        OracleConnection con = new OracleConnection(constr);  
        con.Open();  
  
        // Execute a SQL SELECT  
        OracleCommand cmd = con.CreateCommand();  
        cmd.CommandText = "select * from emp";  
        OracleDataReader reader = cmd.ExecuteReader();  
  
        // Print all employee numbers  
        while (reader.Read())  
            Console.WriteLine(reader.GetInt32(0));  
  
        // Clean up  
        reader.Dispose();  
        cmd.Dispose();  
        con.Dispose();  
    }  
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

Commit()

This method commits the database transaction.

Declaration

```
// C#  
public void Commit();
```

Exception

`InvalidOperationException` – If the connection is closed or if the connection is enlisted in `System.Transactions`.

`System.ObjectDisposedException` – This property cannot be accessed after the `OracleConnection` object is already disposed.

Remarks

This method allows the user to commit an implicit or explicit local transaction or active sessionless transaction using an `OracleConnection` object. Once committed, the transaction ends. For local transactions, it has the same behavior as the `OracleTransaction Commit` method. However, there are instances when this method is necessary to invoke without the `OracleTransaction` object available, such as an implicit transaction using "SELECT ... FOR UPDATE" or an implicit transaction executing an INSERT/UPDATE/DELETE statement with auto-commit disabled.

This method should not encounter any exceptions when the connection is within an explicit local transaction, implicit local transaction, or not associated with a transaction. This method cannot be executed on a connection enlisted in `System.Transactions` (local or distributed).

When an explicit transaction begins, auto-commit is disabled regardless of the `AutoCommit` property value. Once the `Commit` method is invoked, `AutoCommit` reverts back to true if it was earlier enabled. Transaction isolation level will also reset to its earlier value on the `OracleConnection` object upon the commit.

① See Also

- [Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

DisableReplay

This method disables replay on the connection in Application Continuity.

Declaration

```
// C#  
public void DisableReplay();
```

Remarks

By default, ODP.NET replays following a recoverable error. If any application uses a design unsuitable for replay, then this method can be called to disable replay on a per request basis. It can be added to the callback or to the main code. For example, if an application module uses the UTL_SMTP package to send emails, the developer may not want those emails to be sent again upon replay.

Disabling replay does not alter the connection state by re-executing any method, SQL or PL/SQL. When it is disabled, both recording and replay are disabled until that request ends and an explicit BeginRequest is called.

There is no method to re-enable replay once disabled. It is invalid to reestablish the database session with time gaps in a replayed request. This principle ensures replay runs only if a complete call has been recorded.

① See Also

- [Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

EndRequest

This method indicates an ending explicit request boundary when using Application Continuity.

Declaration

```
// C#  
public void EndRequest();
```

Remarks

Using this method demarcates the end of the request boundary and disables replay. A subsequent BeginRequest call will re-establish a request boundary. Existing state on the connection is not affected by this call.

Calling EndRequest multiple times without BeginRequest in between is allowed.

① See Also

- [Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

EnlistDistributedTransaction

This method enables applications to explicitly enlist in a specific distributed transaction after a connection has been opened.

Declaration

```
// C#  
public void EnlistDistributedTransaction(ITransaction transaction);
```

Parameters

- *transaction*
An ITransaction interface.

Exceptions

InvalidOperationException - The connection is part of a local transaction or the connection is closed.

Remarks

EnlistDistributedTransaction enables objects to enlist in a specific transaction that is passed to the method. The ITransaction interface can be obtained by applying an (ITransaction) cast to the ContextUtil.Transaction property within the component that started the distributed transaction.

The connection must be open before calling this method or an InvalidOperationException is thrown.

If a connection is part of a local transaction that was started implicitly or explicitly while attempting to enlist in a distributed transaction, the local transaction is rolled back and an exception is thrown.

By default, distributed transactions roll back, unless the method-level AutoComplete declaration is set.

Invoking the commit on the ITransaction raises an exception.

Invoking the rollback on the ITransaction method and calling ContextUtil.SetComplete on the same distributed transaction raises an exception.

Example

Application:

```
// C#  
  
/* This is the class that will utilize the Enterprise Services  
   component. This module needs to be built as an executable.  
  
   The Enterprise Services Component DLL must be built first  
   before building this module.  
   In addition, the DLL needs to be referenced appropriately  
   when building this application.  
*/  
  
using System;  
using System.EnterpriseServices;
```

```

using DistribTxnSample;

class DistribTxnSample_App
{
    static void Main()
    {
        DistribTxnSample_Comp comp = new DistribTxnSample_Comp();
        comp.DoWork();
    }
}

```

Component:

```

// C#

/* This module needs to be
   1) built as a component DLL/Library
   2) built with a strong name

   This library must be built first before the application is built.
*/

using System;
using System.Data;
using Oracle.DataAccess.Client;
using System.EnterpriseServices;

namespace DistribTxnSample
{
    [Transaction(TransactionOption.RequiresNew)]
    public class DistribTxnSample_Comp : ServicedComponent
    {
        public void DoWork()
        {
            string constr =
                "User Id=scott;Password=tiger;Data Source=oracle;enlist=false";
            OracleConnection con = new OracleConnection(constr);
            con.Open();

            // Enlist in a distributed transaction
            con.EnlistDistributedTransaction((ITransaction)ContextUtil.Transaction);

            // Update EMP table
            OracleCommand cmd = con.CreateCommand();
            cmd.CommandText = "UPDATE emp set sal = sal + .01";
            cmd.ExecuteNonQuery();

            // Commit
            ContextUtil.SetComplete();

            // Dispose OracleConnection object
            con.Dispose();
        }
    }
}

```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- ["EnlistTransaction"](#)
- <http://msdn.microsoft.com/library> for detailed information about this Microsoft .NET Framework feature

EnlistTransaction

This method enlists the connection to the specified transaction.

Declaration

```
// C#  
public override void EnlistTransaction(Transaction transaction)
```

Parameters

- *transaction*
A System.Transactions.Transaction object.

Exceptions

InvalidOperationException - The connection is part of a local transaction or the connection is closed.

Remarks

Invocation of this method immediately enlists the connection to a transaction that is specified by the provided transaction parameter.

If OracleConnection is still associated with a distributed transaction that has not completed from a previous EnlistTransaction method invocation, calling this method will cause an exception to be thrown.

In general, for transaction enlistments to succeed, the "enlist" connection string attribute must be set to "true" before invoking the Open method. Setting the "enlist" connection string attribute to "true" will implicitly enlist the connection when the Open method is called, if the connection is within a transaction context. The "enlist" attribute should be set to "false" or "dynamic" only if the connection will never enlist in a transaction.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- ["System.Transactions and Promotable Transactions"](#)
- ["EnlistDistributedTransaction"](#)

FlushCache

This method flushes all updates and deletes made through REF objects retrieved using this connection.

Declaration

```
// c#  
public void FlushCache();
```

Exceptions

InvalidOperationException - The specified connection is not open.

Remarks

Before flushing objects, it is required that the application has explicitly started a transaction by executing the BeginTransaction method on the OracleConnection object. This is because if the object being flushed has not already been locked by the application, an exclusive lock is obtained implicitly for the object. The lock is only released when the transaction commits or rollbacks.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

GetSchema

GetSchema methods return schema information for the data source of the OracleConnection.

Overload List

- [GetSchema\(\)](#)
This method returns schema information for the data source of the OracleConnection.
- [GetSchema \(string collectionName\)](#)
This method returns schema information for the data source of the OracleConnection using the specified string for the collection name.

- [GetSchema \(string collectionName, string\[\] restrictions\)](#)

This method returns schema information for the data source of the OracleConnection using the specified string for the collection name and the specified string array for the restriction values.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

GetSchema()

This method returns schema information for the data source of the OracleConnection.

Declaration

```
// C#  
public override DataTable GetSchema();
```

Return Value

A DataTable object.

Exceptions

InvalidOperationException – The connection is closed.

Remarks

This method returns a DataTable object that contains a row for each metadata collection available from the database.

The method is equivalent to specifying the String value "MetaDataCollections" when using the GetSchema(String) method.

Example

```
// C#  
  
using System;  
using System.Data;  
using System.Data.Common;  
using Oracle.DataAccess.Client;  
  
class GetSchemaSample  
{  
    static void Main(string[] args)  
    {  
        string constr = "User Id=scott; Password=tiger; Data Source=oracle;";  
        string ProviderName = "Oracle.DataAccess.Client";  
  
        DbProviderFactory factory = DbProviderFactories.GetFactory(ProviderName);
```

```
using (DbConnection conn = factory.CreateConnection())
{
    try
    {
        conn.ConnectionString = constr;
        conn.Open();

        //Get all the schema collections and write to an XML file.
        //The XML file name is Oracle.DataAccess.Client_Schema.xml
        DataTable dtSchema = conn.GetSchema();
        dtSchema.WriteXml(ProviderName + "_Schema.xml");
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
        Console.WriteLine(ex.StackTrace);
    }
}
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

GetSchema (string collectionName)

This method returns schema information for the data source of the OracleConnection using the specified string for the collection name.

Declaration

```
// C#
public override DataTable GetSchema (string collectionName);
```

Parameters

collectionName

Name of the collection for which metadata is required.

Return Value

A DataTable object.

Exceptions

ArgumentException – The requested collection is not defined.

InvalidOperationException – The connection is closed.

InvalidOperationException – The requested collection is not supported by current version of Oracle database.

InvalidOperationException – No population string is specified for requested collection.

Example

```
// C#

using System;
using System.Data;
using System.Data.Common;
using Oracle.DataAccess.Client;

class GetSchemaSample
{
    static void Main(string[] args)
    {
        string constr = "User Id=scott; Password=tiger; Data Source=oracle;";
        string ProviderName = "Oracle.DataAccess.Client";

        DbProviderFactory factory = DbProviderFactories.GetFactory(ProviderName);

        using (DbConnection conn = factory.CreateConnection())
        {
            try
            {
                conn.ConnectionString = constr;
                conn.Open();

                //Get MetaDataCollections and write to an XML file.
                //This is equivalent to GetSchema()
                DataTable dtMetadata =
                    conn.GetSchema(DbMetaDataCollectionNames.MetaDataCollections);
                dtMetadata.WriteXml(ProviderName + "_MetaDataCollections.xml");

                //Get Restrictions and write to an XML file.
                DataTable dtRestrictions =
                    conn.GetSchema(DbMetaDataCollectionNames.Restrictions);
                dtRestrictions.WriteXml(ProviderName + "_Restrictions.xml");

                //Get DataSourceInformation and write to an XML file.
                DataTable dtDataSrcInfo =
                    conn.GetSchema(DbMetaDataCollectionNames.DataSourceInformation);
                dtDataSrcInfo.WriteXml(ProviderName + "_DataSourceInformation.xml");

                //data types and write to an XML file.
                DataTable dtDataTypes =
                    conn.GetSchema(DbMetaDataCollectionNames.DataTypes);
                dtDataTypes.WriteXml(ProviderName + "_DataTypes.xml");

                //Get ReservedWords and write to an XML file.
                DataTable dtReservedWords =
                    conn.GetSchema(DbMetaDataCollectionNames.ReservedWords);
                dtReservedWords.WriteXml(ProviderName + "_ReservedWords.xml");

                //Get all the tables and write to an XML file.
                DataTable dtTables = conn.GetSchema("Tables");
                dtTables.WriteXml(ProviderName + "_Tables.xml");

                //Get all the views and write to an XML file.
                DataTable dtViews = conn.GetSchema("Views");
                dtViews.WriteXml(ProviderName + "_Views.xml");

                //Get all the columns and write to an XML file.
                DataTable dtColumns = conn.GetSchema("Columns");
```



```
        dtColumns.WriteXml(ProviderName + "_Columns.xml");
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
        Console.WriteLine(ex.StackTrace);
    }
}
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

GetSchema (string collectionName, string[] restrictions)

This method returns schema information for the data source of the `OracleConnection` using the specified string for the collection name and the specified string array for the restriction values.

Declaration

```
// C#
public override DataTable GetSchema (string collectionName,
    string[] restrictions);
```

Parameters

- *collectionName*
The name of the collection of metadata being retrieved.
- *restrictions*
An array of restrictions that apply to the metadata being retrieved.

Return Value

A `DataTable` object.

Exception

- `ArgumentException` – The requested collection is not defined.
- `InvalidOperationException` – One of the following conditions exist:
 - The connection is closed.
 - The requested collection is not supported by the current version of Oracle database.
 - More restrictions were provided than the requested collection supports.
 - No population string is specified for requested collection.

Remarks

This method takes the name of a metadata collection and an array of String values that specify the restrictions for filtering the rows in the returned DataTable. This returns a DataTable that contains only rows from the specified metadata collection that match the specified restrictions.

For example, if the Columns collection has three restrictions (owner, tablename, and columnname), to retrieve all the columns for the EMP table regardless of schema, the GetSchema method must pass in at least these values: null, EMP.

If no restriction value is passed in, default values are used for that restriction, which is the same as passing in null. This differs from passing in an empty string for the parameter value. In this case, the empty string ("") is considered the value for the specified parameter.

collectionName is not case-sensitive, but restrictions (string values) are.

Example

```
// C#

using System;
using System.Data;
using System.Data.Common;
using Oracle.DataAccess.Client;

class GetSchemaSample
{
    static void Main(string[] args)
    {
        string constr = "User Id=scott; Password=tiger; Data Source=oracle;";
        string ProviderName = "Oracle.DataAccess.Client";

        DbProviderFactory factory = DbProviderFactories.GetFactory(ProviderName);

        using (DbConnection conn = factory.CreateConnection())
        {
            try
            {
                conn.ConnectionString = constr;
                conn.Open();

                //Get Restrictions
                DataTable dtRestrictions =
                    conn.GetSchema(DbMetaDataCollectionNames.Restrictions);

                DataView dv = dtRestrictions.DefaultView;

                dv.RowFilter = "CollectionName = 'Columns'";
                dv.Sort = "RestrictionNumber";

                for (int i = 0; i < dv.Count; i++)
                    Console.WriteLine("{0} (default) {1}" ,
                                      dtRestrictions.Rows[i]["RestrictionName"],
                                      dtRestrictions.Rows[i]["RestrictionDefault"]);

                //Set restriction string array
                string[] restrictions = new string[3];

                //Get all columns from all tables owned by "SCOTT"
                restrictions[0] = "SCOTT";
                DataTable dtAllScottCols = conn.GetSchema("Columns", restrictions);
            }
        }
    }
}
```

```

// clear collection
for (int i = 0; i < 3; i++)
    restrictions[i] = null;

//Get all columns from all tables named "EMP" owned by any
//owner/schema
restrictions[1] = "EMP";
DataTable dtAllEmpCols = conn.GetSchema("Columns", restrictions);

// clear collection
for (int i = 0; i < 3; i++)
    restrictions[i] = null;

//Get columns named "EMPNO" from tables named "EMP",
//owned by any owner/schema
restrictions[1] = "EMP";
restrictions[2] = "EMPNO";
DataTable dtAllScottEmpCols = conn.GetSchema("Columns", restrictions);

// clear collection
for (int i = 0; i < 3; i++)
    restrictions[i] = null;

//Get columns named "EMPNO" from all
//tables, owned by any owner/schema
restrictions[2] = "EMPNO";
DataTable dtAllEmpNoCols = conn.GetSchema("Columns", restrictions);
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
    Console.WriteLine(ex.Source);
}
}
}
}

```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

GetSessionInfo

GetSessionInfo returns or refreshes an OracleGlobalization object that represents the globalization settings of the session.

Overload List:

- [GetSessionInfo\(\)](#)

This method returns a new instance of the OracleGlobalization object that represents the globalization settings of the session.

- [GetSessionInfo\(OracleGlobalization\)](#)

This method refreshes the provided `OracleGlobalization` object with the globalization settings of the session.

📘 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

GetSessionInfo()

This method returns a new instance of the `OracleGlobalization` object that represents the globalization settings of the session.

Declaration

```
// C#  
public OracleGlobalization GetSessionInfo();
```

Return Value

The newly created `OracleGlobalization` object.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Client;  
  
class GetSessionInfoSample  
{  
    static void Main()  
    {  
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";  
        OracleConnection con = new OracleConnection(constr);  
        con.Open();  
  
        // Get session info from connection object  
        OracleGlobalization info = con.GetSessionInfo();  
  
        // Update session info  
        info.DateFormat = "YYYY-MM-DD";  
        con.SetSessionInfo(info);  
  
        // Execute SQL SELECT  
        OracleCommand cmd = con.CreateCommand();  
        cmd.CommandText = "select TO_CHAR(hiredate) from emp";  
        Console.WriteLine("Hire Date ({0}): {1}",  
            info.DateFormat, cmd.ExecuteScalar());  
  
        // Clean up  
        cmd.Dispose();  
        con.Dispose();  
    }  
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

GetSessionInfo(OracleGlobalization)

This method refreshes the provided OracleGlobalization object with the globalization settings of the session.

Declaration

```
// C#  
public void GetSessionInfo(OracleGlobalization oraGlob);
```

Parameters

- *oraGlob*

The OracleGlobalization object to be updated.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

Open

This method opens a connection to an Oracle database.

Declaration

```
// C#  
public override void Open();
```

Implements

IDbConnection

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The connection is already opened or the connection string is null or empty.

OracleException – For token authentication, errors include "ORA-25707: The token is invalid.", "ORA-25708: The token has expired.", and "ORA-50109: Configuration for token-based authentication is invalid."

Remarks

The connection is obtained from the pool if connection pooling is enabled. Otherwise, a new connection is established.

It is possible that the pool does not contain any unused connections when the `Open()` method is invoked. In this case, a new connection is established.

If no connections are available within the specified connection timeout value, when the Max Pool Size is reached, an `OracleException` is thrown.

For client-side validation of an access token:

- The access token must be 16KB bytes or less.
- There needs to be two periods, ".".
- It must be valid base 64.
- It must be valid JSON.
- There must be "sub" and "exp" claims.

The opened connection will have same `AutoCommit` property value as set by the application.

If the opened connection enlists in a distributed transaction, auto-commit will be disabled regardless of the `AutoCommit` value. Once that transaction completes, auto-commit functionality resets to the `AutoCommit` value.

The opened connection has the same `IsolationLevel` value as set by the application. If that connection enlists in a distributed transaction created using `TransactionScope`, then the application's `IsolationLevel` value is overridden. The connection object's `IsolationLevel` remains overridden once that transaction is over.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

OpenAsync

`OpenAsync` returns a Task-based asynchronous version of `OracleConnection.Open`.

Overload List:

- [OpenAsync\(\)](#)

This method calls into the `OpenAsync` implementation with argument `cancellation_token` passed as `CancellationToken.None`.

- [OpenAsync\(CancellationToken cancellation_token\)](#)

This method returns a Task object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of opening connection.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

OpenAsync()

This method will call into the OpenAsync implementation with argument cancellation token passed as Cancellation token.None.

Declaration

```
// C#  
public Task OpenAsync();
```

Return Value

Task object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of opening connection.

Implements

DbConnection

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `InvalidOperationException` - The connection is already opened or the connection string is null or empty.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

OpenAsync(CancellationToken cancellationToken)

This method returns a new Task-based asynchronous version of `OracleConnection.Open`.

Declaration

```
// C#  
public override Task OpenAsync(CancellationToken cancellationToken);
```

Parameters

- `cancellationToken` - The input cancellation token which can be used by the application to cancel the task before connection timeout occurs.

Return Value

Task object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of opening connection.

Implements

DbConnection

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `InvalidOperationException` - The connection is already opened or the connection string is null or empty.

Remarks

- After calling `OpenAsync`, `OracleConnection.State` will return `Connecting` until the returned Task is completed. Then, if the connection was successful, State will return `Open`. If the connection fails, State will return `Closed`.
- If the connection timeout time elapses without successfully connecting, the returned Task will be marked as faulted with an Exception.

Example

```
using Oracle.ManagedDataAccess.Client;
using System;
using System.Threading;
using System.Threading.Tasks;

namespace AsyncApp
{
    class AsyncDemo
    {
        static async Task Main()
        {
            string connectionString = "User Id=HR; Password=<PASSWORD>; Data Source=oracle;";
            OracleConnection oc = new OracleConnection(connectionString);

            // Open a connection asynchronously
            Task task = oc.OpenAsync(CancellationToken.None);

            // Execute an operation while the connection is being opened asynchronously
            Console.WriteLine("Opening a connection asynchronously.");

            // wait for the connection to be opened
            await task;
            Console.WriteLine("Connection opened successfully");
            oc.Close();
        }
    }
}
```


① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

OpenWithNewClientSecret

This method opens a new connection using a new Microsoft Entra ID client secret.

Declaration

```
// C#  
public static void OpenWithNewClientSecret(this OracleConnection @this, SecureString  
clientSecret)
```

Parameters

clientSecret – new client secret SecureString object reference

Exceptions

- `ArgumentNullException` is raised if new client secret is passed in as null.
- `ArgumentException` is raised if new client secret length is 0 or if corresponding secure string is not set as readonly.
- `InvalidOperationException` - The connection is already opened or `TokenAuthentication` property is not set to `AzureServicePrincipal`.
- `OracleException` – For token authentication, errors include "ORA-50109: Configuration for token-based authentication is invalid"

Description

ODP.NET uses the new client secret value to open a new connection. If the new connection opens successfully, then this new client secret is cached with the connection pool. If the new connection fails to open successfully, then an exception is thrown and the new client secret is ignored.

Remarks

This method is recommended only for when the current Azure client secret expires or opening a connection with a different client secret.

ODP.NET does not check whether the new client secret is the same value or reference as the existing cached client secret in the connection pool.

This method should successfully open an entirely new connection and connection pool even if there is no existing connection pool associated.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

OpenWithNewPassword

OpenWithNewPassword opens a new connection with the new password.

Overload List:

- [OpenWithNewPassword\(String\)](#)
OpenWithNewPassword opens a new connection with the new password.
- [OpenWithNewPassword\(SecureString\)](#)
OpenWithNewPassword opens a new connection with the new password. This method will also update the password in the OracleCredential instance that was passed using the OracleConnection constructor or through the OracleConnection Credential property.
- [OpenWithNewPassword\(OracleOpaqueString\)](#)
OpenWithNewPassword opens connection with a new OracleOpaqueString password.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

OpenWithNewPassword(String)

This method opens a new connection with the new password.

Declaration

```
// C#  
public void OpenWithNewPassword(string newPassword);
```

Parameters

- *newPassword*
A string that contains the new password.

Remarks

This method uses the ConnectionString property settings to establish a new connection. The old password must be provided in the connection string as the Password attribute value.

This method can only be called on an OracleConnection in the *closed* state.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- ["Password Expiration"](#)

OpenWithNewPassword(SecureString)

This method opens a new connection with the new password.

Declaration

```
// C#  
public void OpenWithNewPassword(SecureString newSecurePassword)
```

Parameters

- *newSecurePassword*

The SecureString object that contains the new password for a user id.

Remarks

Use this method to change the password for a user id. This method can only be used if an OracleCredential instance was passed using the OracleConnection constructor or through the OracleConnection Credential property.

This method will also update the password in the OracleCredential instance that was passed using the OracleConnection constructor or through the OracleConnection Credential property.

OracleCredential does not support double quotes around a SecureString password. Double quotes can be used within a password, however.

Exceptions

ArgumentNullException is raised if *newSecurePassword* is NULL or is not read only.

InvalidOperationException is raised if OracleCredential is not already set on the OracleConnection either through the OracleConnection constructor or through the Credential property of the OracleConnection.

InvalidOperationException is raised if the OracleConnection is already in the open state.

InvalidOperationException is raised if the connection string is NULL or of length 0.

InvalidOperationException is raised if the user id, proxy user id, password, proxy password, DBA privilege is set using the connection string.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- ["Password Expiration"](#)

OpenWithNewPassword(OracleOpaqueString)

This method opens connection with a new `OracleOpaqueString` password.

Declaration

```
// C#  
public void OpenWithNewPassword(OracleOpaqueString newOpaquePassword)
```

Parameters

- *newOpaquePassword*
The new password to open a connection with.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

OpenWithNewPasswordAsync

`OpenWithNewPasswordAsync` is a Task-based asynchronous version of `OracleConnection.OpenWithNewPassword()`, which opens a new connection with the new password.

Overload List:

- [OpenWithNewPasswordAsync\(String\)](#)
Opens a new connection asynchronously with a new password.
- [OpenWithNewPasswordAsync\(string, CancellationToken\)](#)
Opens a new connection asynchronously with a new password and token that can cancel the task.
- [OpenWithNewPasswordAsync\(SecureString\)](#)
Opens a new connection asynchronously with a new `SecureString` password.
- [OpenWithNewPasswordAsync\(SecureString, CancellationToken\)](#)
Opens a new connection asynchronously with a new `SecureString` password and token that can cancel the task.

- [OpenWithNewPasswordAsync\(OracleOpaqueString\)](#)
Asynchronously opens connection with a new `OracleOpaqueString` password
- [OpenWithNewPasswordAsync\(OracleOpaqueString, CancellationToken\)](#)
Asynchronously opens connection with a new `OracleOpaqueString` password and a `CancellationToken` to cancel thread.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

OpenWithNewPasswordAsync(String)

This method opens a new connection asynchronously with a new password.

Declaration

```
// C#  
public Task OpenWithNewPasswordAsync(string newPassword);
```

Parameters

- *newPassword*
A string that contains the new password.

Return Value

Task object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of opening connection.

Remarks

This method uses the `ConnectionString` property settings to establish a new connection. The old password must be provided in the connection string as the `Password` attribute value.

This method can only be called on an `OracleConnection` in the closed state.

Example

```
namespace AsyncApp  
{  
    class AsyncDemo  
    {  
        static async Task Main()  
        {  
            //Specify old password in connection string  
            string connectionString = "User Id=HR; Password=<PASSWORD>; Data Source=oracle;";  
            OracleConnection oc = new OracleConnection(connectionString);  
  
            String newPass = "newPass";  
  
            // Open a connection asynchronously with new password
```

```
Task task = oc.OpenWithNewPasswordAsync(newPass);

// Execute an operation while the connection is being opened asynchronously
Console.WriteLine("Opening a connection asynchronously.");

// wait for the connection to be opened
await task;
Console.WriteLine("Connection opened successfully, password changed to " +
newPass);
    oc.Close();
}
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- ["Password Expiration"](#)

OpenWithNewPasswordAsync(string, CancellationToken)

This method opens a new connection asynchronously with a new password and token that can cancel the task.

Declaration

```
// C#
public Task OpenWithNewPasswordAsync(string newPassword, CancellationToken
cancellation_token);
```

Parameters

- *newPassword*
A string that contains the new password.
- *cancellation_token*
The input cancellation token which can be used by the application to cancel the task before connection timeout occurs.

Return Value

Task object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of opening connection.

Remarks

This method uses the `ConnectionString` property settings to establish a new connection. The old password must be provided in the connection string as the `Password` attribute value.

This method can only be called on an `OracleConnection` in the closed state.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- ["Password Expiration"](#)

OpenWithNewPasswordAsync(SecureString)

This method opens a new connection asynchronously with a new SecureString password.

Declaration

```
// C#  
public Task OpenWithNewPasswordAsync(SecureString newSecurePassword);
```

Parameters

- *newSecurePassword*
A SecureString object that contains the new password for a user id.

Return Value

Task object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of opening connection.

Exceptions

- `ArgumentNullException` - `newSecurePassword` is NULL or is not read only.
- `InvalidOperationException` - `OracleCredential` is not already set on the `OracleConnection` either through the `OracleConnection` constructor or through the `Credential` property of the `OracleConnection`.
- `InvalidOperationException` - the `OracleConnection` is already in the open state.
- `InvalidOperationException` - the connection string is NULL or of length 0.
- `InvalidOperationException` - the user id, proxy user id, password, proxy password, DBA privilege is set using the connection string.

Remarks

Use this method to change the password for a user id. This method can only be used if an `OracleCredential` instance was passed using the `OracleConnection` constructor or through the `OracleConnection.Credential` property.

This method will also update the password in the `OracleCredential` instance that was passed using the `OracleConnection` constructor or through the `OracleConnection.Credential` property.

`OracleCredential` does not support double quotes around a `SecureString` password. However, double quotes can be used within a password.

① See Also

- [Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- ["Password Expiration"](#)

OpenWithNewPasswordAsync(SecureString, CancellationToken)

This method opens a new connection asynchronously with a new SecureString password and token that can cancel the task.

Declaration

```
// C#  
public Task OpenWithNewPasswordAsync(SecureString newSecurePassword, CancellationToken  
    cancellationToken);
```

Parameters

- *newSecurePassword*
A SecureString object that contains the new password for a user id.
- *cancellationToken*
The input cancellation token which can be used by the application to cancel the task before connection timeout occurs.

Return Value

Task object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of opening connection.

Exceptions

- *ArgumentNullException* - newSecurePassword is NULL or is not read only.
- *InvalidOperationException* - OracleCredential is not already set on the OracleConnection either through the OracleConnection constructor or through the Credential property of the OracleConnection.
- *InvalidOperationException* - the OracleConnection is already in the open state.
- *InvalidOperationException* - the connection string is NULL or of length 0.
- *InvalidOperationException* - the user id, proxy user id, password, proxy password, DBA privilege is set using the connection string.

Remarks

Use this method to change the password for a user id. This method can only be used if an OracleCredential instance was passed using the OracleConnection constructor or through the OracleConnection Credential property.

This method will also update the password in the `OracleCredential` instance that was passed using the `OracleConnection` constructor or through the `OracleConnection.Credential` property.

`OracleCredential` does not support double quotes around a `SecureString` password. However, double quotes can be used within a password.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- ["Password Expiration"](#)

OpenWithNewPasswordAsync(OracleOpaqueString)

This method asynchronously opens connection with a new `OracleOpaqueString` password.

Declaration

```
// C#  
public void OpenWithNewPasswordAsync(OracleOpaqueString newOpaquePassword)
```

Parameters

- *newOpaquePassword*
The new password to open a connection with.

See Also

- [Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

OpenWithNewPasswordAsync(OracleOpaqueString, CancellationToken)

This method asynchronously opens connection with a new `OracleOpaqueString` password.

Declaration

```
// C#  
public void OpenWithNewPasswordAsync(OracleOpaqueString newOpaquePassword,  
CancellationToken token)
```

Parameters

- *newOpaquePassword*
The new password to open a connection with.

- *token*

Token to cancel thread.

① See Also

- [Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

OpenWithNewToken

OpenWithNewToken opens a new connection by providing the refreshed/updated access token directly to ODP.NET.

Overload List:

- [OpenWithNewToken\(char\[\]\)](#)

OpenWithNewToken(char[]) opens a new connection by providing the refreshed/updated access token directly to ODP.NET instead of through the token refresh call back.

- [OpenWithNewToken\(char\[\], char\[\]\)](#)

OpenWithNewToken(char[], char[]) opens a new connection by providing the refreshed/updated signature token and private key directly to ODP.NET instead of through the token refresh call back

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

OpenWithNewToken(char[])

This method can be used to open a new connection by providing the refreshed/updated access token directly to ODP.NET instead of through the token refresh call back. It is recommended to be used only in cases the application is unable or fails to provide the refreshed/updated access token through the token refresh call back.

Declaration

```
// C#  
public void OpenWithNewToken(char[] token)
```

Exceptions

- `ArgumentNullException` is raised if token is passed in as null.

- `InvalidOperationException` - The connection is already opened, or there is not an `OracleAccessToken` object set on the connection.
- `OracleException` – For token authentication, errors include “ORA-25707: The token is invalid.”, and “ORA-25708: The token has expired.”

Description

ODP.NET uses the provided access token to open a new database connection. If the new connection is opened successfully, then the underlying `OracleAccessToken` object is updated with the new access token. If the driver fails to open the new connection, then an exception is thrown and the access token is ignored.

The `char[]` passed for the access token will be cleared ODP.NET for security reasons. The application should not depend on the `char[]` once this method is called.

For client-side validation of an access token:

- The access token must be 16KB bytes or less.
- There needs to be two periods, ".".
- It must be valid base 64.
- It must be valid JSON.
- There must be "sub" and "exp" claims.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

OpenWithNewToken(char[], char[])

This method can be used by the application to open a new connection by providing the refreshed/updated signature token and private key directly to ODP.NET instead of through the token refresh call back. It is recommended to be used only in cases the application is unable or fails to provide the refreshed/updated token and private key through the token refresh call back.

Declaration

```
// C#  
public void OpenWithNewToken(char[] dbToken, char[] privateKey)
```

Exceptions

- `ArgumentNullException` is raised if any parameters are passed in as null.
- Exception if connection is already open.
- Exception if the 'sub' value of the new token does not match the 'sub' value of the current token.

Description

The 'sub' value of the new token must be the same as the 'sub' value of the current token. ODP.NET will use the application provided signature token and private key to open a new connection to the database. If the new connection can be opened successfully, then the underlying `OracleAccessToken` object will be updated with these new token and private key. If ODP.NET fails to open the new connection, then an exception will be thrown to the application and the `OracleAccessToken` object will not be updated with the new token and private key.

The `char[]` passed by the application for DB token and private key will be cleared by ODP.NET. Applications should not depend on these `char[]` once this constructor is called.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

PurgeStatementCache

This method flushes the statement cache by closing all open cursors on the database, when statement caching is enabled.

Declaration

```
// C#  
public void PurgeStatementCache();
```

Remarks

Flushing the statement cache repetitively results in decreased performance and may negate the performance benefit gained by enabling the statement cache.

Statement caching remains enabled after the call to `PurgeStatementCache`.

Invocation of this method purges the cached cursors that are associated with the `OracleConnection`. It does not purge all the cached cursors in the database.

Example

```
// C#  
  
using System;  
using System.Data;  
using Oracle.DataAccess.Client;  
  
class PurgeStatementCacheSample  
{  
    static void Main()  
    {  
        string constr = "User Id=scott;Password=tiger;Data Source=oracle;" +  
            "Statement Cache Size=20";  
        OracleConnection con = new OracleConnection(constr);  
        con.Open();  
  
        OracleCommand cmd = new OracleCommand("select * from emp", con);
```

```
cmd.CommandType = CommandType.Text;
OracleDataReader reader = cmd.ExecuteReader();

// Purge Statement Cache
con.PurgeStatementCache();

// Close and Dispose OracleConnection object
Console.WriteLine("Statement Cache Flushed");
con.Close();
con.Dispose();
}
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- ["Statement Caching"](#)
- [ConnectionString](#)

ResumeSessionlessTransaction

This method resumes a sessionless transaction using the provided unique transaction identifier.

Declaration

```
// C#
public void ResumeSessionlessTransaction(byte[] txnID);
```

Parameters

- `txnID`
the sessionless transaction identifier.

Exceptions

`InvalidOperationException` - The connection is already associated with a transaction or is currently in a transaction.

`ArgumentException` - The specified sessionless transaction identifier is invalid.

`OracleException` - The provided sessionless transaction identifier does not exist in the database (ORA-26218).

`OracleException` - The database does not support sessionless transactions (ORA-26207).

Remarks

The transaction identifier can be obtained from the `BeginSessionlessTransaction()` method return value. `OracleConnection` can operate on a sessionless transaction through its `Commit()`, `Rollback()`, and `SuspendSessionlessTransaction()` methods.

Auto-commit is disabled when this method is invoked successfully.

If the `OracleConnection.SessionlessTransactionOptions` is set, then `ResumeSessionlessTransaction()` operates the sessionless transaction based upon the `OracleConnection.SessionlessTransactionOptions` specified settings.

If the `OracleConnection.SessionlessTransactionOptions` is not set, then `ResumeSessionlessTransaction()` operates the sessionless transaction using the following default behavior:

- Sessionless transaction timeout is 60 seconds.
- The connection is associated with the `OracleConnection.ResumeSessionlessTransaction()` sessionless transaction. The transaction only resumes upon the first `OracleCommand` SQL execution.
- The sessionless transaction will not be suspended implicitly after SQL execution.

See Also

- [Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

Rollback

Rollback rolls back the local transaction.

Overload List:

- [Rollback\(\)](#)
This method rolls back the local transaction.
- [Rollback\(String\)](#)
This method rolls back the local transaction to a specified savepoint.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

Rollback()

This method rolls back the local transaction.

Declaration

```
// C#  
public void Rollback();
```

Exception

`InvalidOperationException` – If the connection is closed or if the connection is enlisted in `System.Transactions` (both local and distributed).

`System.ObjectDisposedException` – This property cannot be accessed after the `OracleConnection` object is already disposed.

Remarks

This method allows the user to rollback a local transaction or sessionless transaction using an `OracleConnection` object. Once rolled back, the transaction ends. For local transactions, it has the same behavior as the `OracleTransaction.Rollback` method. However, there are instances when this method is necessary to invoke without the `OracleTransaction` object available, such as an implicit transaction using “SELECT ... FOR UPDATE” or an implicit transaction executing an INSERT/UPDATE/DELETE statement with auto-commit disabled.

This method should not encounter any exceptions when the connection is within an explicit local transaction, implicit local transaction, or not associated with a transaction. This method cannot be executed on a connection enlisted in `System.Transactions` (local or distributed).

When an explicit transaction begins, auto-commit is disabled regardless of the `AutoCommit` property value. Once the `Rollback` method is invoked, `AutoCommit` reverts back to true if it was earlier enabled. Transaction isolation level will also reset to its earlier value on the `OracleConnection` object upon the rollback.

① See Also

- [Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

Rollback(String)

This method rolls back the local transaction to a specified savepoint.

Declaration

```
// C#  
public void Rollback(string savepointName);
```

Exception

`InvalidOperationException` – If the connection is closed or if the connection is enlisted in `System.Transactions` (both local and distributed).

`System.ObjectDisposedException` – This property cannot be accessed after the `OracleConnection` object is already disposed.

`Oracle.ManagedDataAccess.Client.OracleException` – When a savepoint passed in has not been previously established by a `Save(String savepoint)`, or is an invalid savepoint.

Remarks

After a rollback to a savepoint, the current transaction remains active and can be used for further operations.

This method allows the application to rollback a local transaction to the specific savepoint. It has the same behavior as the `OracleTransaction.Rollback(string savepoint)` method. However, there are instances when this method is necessary to invoke without the `OracleTransaction` object available, such as an implicit transaction using "SELECT ... FOR UPDATE" or an implicit transaction executing an INSERT/UPDATE/DELETE statement with auto-commit disabled.

This method should not encounter any exceptions when the connection is within an explicit local transaction, implicit local transaction, or not associated with a transaction. This method cannot be executed on a connection enlisted in `System.Transactions`.

Since savepoints are created in the database in a case-insensitive manner, the savepoints name cases of the Rollback and Save method do not have to match.

See Also

- [Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

Save(String)

This method creates a savepoint in the database.

Declaration

```
// C#  
public void Save(string savePointName);
```

Parameters

savePointName - The name of the savepoint being created in the current transaction.

Exception

`InvalidOperationException` – The transaction has already been completed successfully, has been rolled back, the connection is closed, or if the connection is enlisted in `System.Transactions` (local or distributed).

`System.ObjectDisposedException` – This property cannot be accessed after the `OracleConnection` object is already disposed.

Remarks

After creating a savepoint, the transaction does not enter a completed state. It can be rolled back to the specific savepoint instead of rolling back the entire transaction

This method allows the user to create a savepoint within a transaction. It has the same behavior as the `OracleTransaction.Save(string savepoint)` method. However, there are instances when this method is necessary to invoke without the `OracleTransaction` object

available, such as an implicit transaction using "SELECT ... FOR UPDATE" or an implicit transaction executing an INSERT/UPDATE/DELETE statement with auto-commit disabled.

This method should not encounter any exceptions when the connection is within an explicit local transaction, implicit local transaction, or not associated with a transaction. This method cannot be executed on a connection enlisted in System.Transactions.

If a previously used savepoint name is specified in the same transaction, the previous one is overwritten with the current savepoint.

Since savepoints are created in the database in a case-insensitive manner, the savepoint name cases of Rollback and Save methods do not have to match.

See Also

- [Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

SetSessionInfo

This method alters the session's globalization settings with all the property values specified in the provided OracleGlobalization object.

Declaration

```
// C#  
public void SetSessionInfo(OracleGlobalization oraGlob);
```

Parameters

- *oraGlob*

An OracleGlobalization object.

Remarks

Calling this method is equivalent to calling an ALTER SESSION SQL on the session.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Client;  
  
class SetSessionInfoSample  
{  
    static void Main()  
    {  
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";  
        OracleConnection con = new OracleConnection(constr);  
        con.Open();  
  
        // Get session info from connection object  
        OracleGlobalization info = con.SetSessionInfo();  
    }  
}
```

```
// Execute SQL SELECT
OracleCommand cmd = con.CreateCommand();
cmd.CommandText = "select TO_CHAR(hiredate) from emp";
Console.WriteLine("Hire Date ({0}): {1}",
    info.DateFormat, cmd.ExecuteScalar());

// Update session info
info.DateFormat = "MM-DD-RR";
con.SetSessionInfo(info);

// Execute SQL SELECT again
Console.WriteLine("Hire Date ({0}): {1}",
    info.DateFormat, cmd.ExecuteScalar());

// Clean up
cmd.Dispose();
con.Dispose();
}
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

SetShardingKey(OracleShardingKey, OracleShardingKey)

This instance method enables applications to set the sharding key and the super sharding key before requesting a connection.

Declaration

```
// C#
public void SetShardingKey(OracleShardingKey shardKey, OracleShardingKey
superShardingKey);
```

Exceptions

InvalidArgumentException – An invalid Oracle sharding key is supplied.

InvalidOperationException – The method is invoked when the connection is in an Open state.

Remarks

This method sets the sharding key and the super sharding key that is to be used for returning the proper connection upon the Open method invocation.

This method can only be invoked when the connection is in a Closed state.

Example

```
// C#

using System;
using Oracle.DataAccess.Client;
```

```
class Sharding
{
    static void Main()
    {
        OracleConnection con = new OracleConnection("user id=hr;password=hr;Data
Source=orcl;");
        //Setting a shard key
        OracleShardingKey shardingKey = new OracleShardingKey(OracleDbType.Int32, 123);
        //Setting a second shard key value for a composite key
        shardingKey.SetShardingKey(OracleDbType.Varchar2, "gold");
        //Creating and setting the super shard key
        OracleShardingKey superShardingKey = new OracleShardingKey();
        superShardingKey.SetShardingKey(OracleDbType.Int32, 1000);

        //Setting super sharding key and sharding key on the connection
        con.SetShardingKey(shardingKey, superShardingKey);
        con.Open();

        //perform SQL query
    }
}
```

📘 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- [OracleShardingKey Class](#)

SuspendSessionlessTransaction

This method suspends the active sessionless transaction immediately from the session.

Declaration

```
// C#
public void SuspendSessionlessTransaction();
```

Exceptions

OracleException - The active transaction is not a sessionless transaction (ORA-26202).

Remarks

If the connection is associated with the active sessionless transaction, the transaction is suspended immediately and will not be associated with the session.

If the connection is not associated with a sessionless transaction, it is a no-op.

① See Also

- [Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

UseAzureTokenAuthentication

This method associates the Microsoft Entra ID token authentication settings with the `OracleConnection` object for use upon the next connection open.

Declaration

```
// C#  
public static void UseAzureTokenAuthentication(this OracleConnection @this,  
AzureTokenAuthentication azureTokenAuthentication = null)
```

Parameters

azureTokenAuthentication – `AzureTokenAuthentication` object reference. Default value is `null`.

Description

This method accepts an `AzureTokenAuthentication` object reference, which ODP.NET uses for access token retrieval upon opening a connection. Once opened, the object becomes read-only and cannot be modified further.

If multiple connections use the same `AzureTokenAuthentication` object reference, connection string, and pool identifiers, then all those connections will be part of the same connection pool.

If the `AzureTokenAuthentication` object reference is omitted, ODP.NET only uses token parameters configured through `tnsnames.ora`, Easy Connect Plus, `sqlnet.ora`, SEPS wallet, and/or Azure .NET SDK environment variables. These parameters values then act as connection pool identifiers.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

UseOciTokenAuthentication

This method associates the OCI IAM token authentication settings with the `OracleConnection` object for use upon the next connection open.

Declaration

```
// C#  
public static void UseOciTokenAuthentication(this OracleConnection @this,  
OciTokenAuthentication ociTokenAuth);
```

Remarks

The object hashcode passed in this method is part of the connection pool manager identification so that different objects result in different connection pools.

Once the ODP.NET OCI token authentication object opens a connection successfully, it becomes read only and can no longer be modified.

Exceptions

- If the connection is already open
- TokenAuthentication property is set to any value other than OciApiKey, OciInstancePrincipal, OciDelegationToken, OciInteractive, or OciDefault.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- ["Statement Caching"](#)
- [ConnectionString](#)

OracleConnection Events

OracleConnection events are listed in [Table 6-54](#).

Table 6-54 OracleConnection Events

Event Name	Description
ConnectionOpen	This event is triggered upon the OracleConnection.Open() method
Disposed	Inherited from System.ComponentModel.Component
Failover	An event that is triggered when an Oracle failover occurs <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
HAEvent	An event that is triggered when an HA event occurs
InfoMessage	An event that is triggered for any message or warning sent by the database
StateChange	An event that is triggered when the connection state changes

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)

ConnectionOpen

This event is triggered upon the `OracleConnection.Open()` method or a connection opening to undertake Application Continuity failover and replay.

Declaration

```
// C#  
public event OracleConnectionOpenEventHandler ConnectionOpen;
```

Event Data

The event handler receives a `OracleConnectionOpenEventArgs` object which exposes the following property containing information about the `ConnectionOpen` event.

- `Connection`
`OracleConnection` object on which `Open()` is called.

Exceptions

- `InvalidOperationException()` - if `CPVersion=1.0` and the `ConnectionOpen` event is used. Applies to unmanaged ODP.NET only.
- `InvalidOperationException()` - if the `ConnectionOpen` event is set after opening a connection.

Remarks

This feature requires `CPVersion=2.0` to be used.

In order to configure the connection before it is dispensed, the application should register the callback to the `ConnectionOpen` event before `Open()` is called.

Only supported for .NET Framework 4.x and .NET (Core).

Application Continuity Scenarios

For applications that set state only at the beginning of a request or for stateful applications that gain performance benefits from using connections with a preset state, they can register for a `ConnectionOpen` event and then check the reason for the event before taking appropriate action.

When registered, the event will be executed upon successful reconnection following a recoverable error. The triggering event reason will be available through `ConnectionOpenReason` property of `OracleConnectionOpenEventArgs` object provided with the event.

The application is responsible for ensuring the initialization actions are the same as that of the original connection before failover. If the event invocation fails, then replay is disabled on that connection.

Example

```
// C#  
// NOTE: The sample below requires CPVersion=2.0 to be configured in the .NET  
configuration  
using System;  
using Oracle.ManagedDataAccess.Client;
```

```
class ConOpenEventSample
{
    public static void ConOpenCallback(OracleConnectionOpenEventArgs eventArgs)
    {
        OracleCommand cmd = new OracleCommand("ALTER SESSION SET
NLS_LANGUAGE='GERMAN'", eventArgs.Connection);
        cmd.ExecuteNonQuery();
        cmd.Dispose();
    }

    static void Main(string[] args)
    {
        // Establish a connection
        string constr = "user id=hr;password=hr;data source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.ConnectionOpen += ConOpenCallback;
        con.Open();
        con.Dispose();
    }
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- [OracleConnectionOpenEventArgs Properties](#)
- [OracleConnectionOpenEventHandler Delegate](#)

Failover

This event is triggered when an Oracle failover occurs.

Declaration

```
// C#
public event OracleFailoverEventHandler Failover;
```

Event Data

The event handler receives an `OracleFailoverEventArgs` object which exposes the following properties containing information about the event.

- `FailoverType`
Indicates the type of the failover.
- `FailoverEvent`
Indicates the state of the failover.

Remarks

The Failover event is raised when a connection to an Oracle instance is unexpectedly severed. The client should create an `OracleFailoverEventHandler` delegate to listen to this event.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- ["OracleFailoverEventArgs Properties"](#)
- ["OracleFailoverEventHandler Delegate"](#)

HAEvent

This event is triggered when an HA event occurs.

Declaration

```
// C#  
public static event OracleHAEventHandler HAEvent;
```

Event Data

The event handler receives an `OracleHAEventArgs` object which exposes the following properties containing information about the event.

- **Source**
Indicates the source of the event.
- **Status**
Indicates the status of the event.
- **DatabaseName**
Indicates the database name affected by this event.
- **DatabaseDomainName**
Indicates the database domain name affected by this event.
- **HostName**
Indicates the host name affected by this event.
- **InstanceName**
Indicates the instance name affected by this event.
- **ServiceName**
Indicates the service name affected by this event.
- **Time**
Indicates the time of the event.

Remarks

The HAEvent is static, which means that any HA Events that happen within the application domain can trigger this event. Note that in order to receive HA event notifications, OracleConnection objects that establish connections within the application domain must have "ha_events=true" in the application. Otherwise, the application never receives any HA Events.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- ["OracleHAEventArgs Properties"](#)
- ["OracleHAEventHandler Delegate"](#)

InfoMessage

This event is triggered for any message or warning sent by the database.

Declaration

```
// C#  
public event OracleInfoMessageEventHandler InfoMessage;
```

Event Data

The event handler receives an OracleInfoMessageEventArgs object which exposes the following properties containing information about the event.

- Errors
The collection of errors generated by the data source.
- Message
The error text generated by the data source.
- Source
The name of the object that generated the error.

Remarks

In order to respond to warnings and messages from the database, the client should create an OracleInfoMessageEventHandler delegate to listen to this event.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- ["OracleInfoMessageEventArgs Properties"](#)
- ["OracleInfoMessageEventHandler Delegate"](#)

StateChange

This event is triggered when the connection state changes.

Declaration

```
// C#  
public override event StateChangeEventHandler StateChange;
```

Event Data

The event handler receives a `StateChangeEventArgs` object which exposes the following properties containing information about the event.

- `CurrentState`
The new state of the connection.
- `OriginalState`
The original state of the connection.

Remarks

The `StateChange` event is raised after a connection changes state, whenever an explicit call is made to `Open`, `Close` or `Dispose`.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnection Class](#)
- [OracleConnection Members](#)
- Microsoft ADO.NET documentation for a description of `StateChangeEventHandler`

OracleConnectionOpenEventArgs Class

The `OracleConnectionOpenEventArgs` class provides connection information for the `OracleConnection.Open()` method. When a connection is opened, the callback registered using `ConnectionOpen` is triggered along with the `OracleConnectionOpenEventArgs` object.

Class Inheritance

System.Object

System.EventArgs

Oracle.DataAccess.Client.OracleConnectionOpenEventArgs

Declaration

```
// C#
public sealed class OracleConnectionOpenEventArgs
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleConnectionOpenEventArgs Members](#)
- [OracleConnectionOpenEventArgs Properties](#)

OracleConnectionOpenEventArgs Members

OracleConnectionOpenEventArgs members are listed in the following tables.

OracleConnectionOpenEventArgsProperties

OracleConnectionOpenEventArgs properties are listed in [Table 6-55](#).

Table 6-55 OracleConnectionOpenEventArgs Properties

Property	Description
Connection	Specifies the OracleConnection on which connection open event has occurred
ConnectionOpenReason	Indicates the reason triggering the ConnectionOpen event <i>Not available in ODP.NET, Unmanaged Driver</i>

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionOpenEventArgs Class](#)

OracleConnectionOpenEventArgs Properties

OracleConnectionOpenEventArgs properties are listed in [Table 6-56](#).

Table 6-56 OracleConnectionOpenEventArgs Properties

Property	Description
Connection	Specifies the OracleConnection on which connection open event has occurred
ConnectionOpenReason	Indicates the reason triggering the ConnectionOpen event

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionOpenEventArgs Class](#)
- [OracleConnectionOpenEventArgs Members](#)

Connection

This property specifies the OracleConnection on which connection open event has occurred.

Declaration

```
// C#  
public Connection{ get;}
```

Property Value

Returns a OracleConnection object on which Open() is called.

Remarks

Only supported for .NET Framework 4.x and .NET (Core).

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionOpenEventArgs Class](#)
- [OracleConnectionOpenEventArgs Members](#)

ConnectionOpenReason

This property indicates the reason triggering the ConnectionOpen event.

Declaration

```
// C#
public OracleConnectionOpenReason ConnectionOpenReason { get; }
```

Property Value

Returns OracleConnectionOpenReason enumeration value indicating the ConnectionOpen event triggering reason.

📘 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionOpenEventArgs Class](#)
- [OracleConnectionOpenEventArgs Members](#)

OracleConnectionOpenEventHandler Delegate

The OracleConnectionOpenEventHandler delegate represents the signature of the method that handles OracleConnection.ConnectionOpen event.

Declaration

```
// C#
public delegate void OracleConnectionOpenEventHandler (OracleConnectionOpenEventArgs
eventArgs);
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Parameters

- *eventArgs*
The OracleConnectionOpenEventArgs object that contains the connection data.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [ConnectionOpen](#)

OracleConnectionStringBuilder Class

An OracleConnectionStringBuilder object allows applications to create or modify connection strings.

Class Inheritance

System.Object
 System.Data.Common.DbConnectionStringBuilder
 Oracle.DataAccess.Client.OracleConnectionStringBuilder

Declaration

```
// C#  
public sealed class OracleConnectionStringBuilder : DbConnectionStringBuilder
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

The following rules must be followed for setting values with reserved characters:

1. Values containing characters enclosed within single quotes
If the value contains characters that are enclosed within single quotation marks, then the entire value must be enclosed within double quotation marks.
For example, password = "' scOTT'" where the value is 'scOTT'.
2. Values containing characters enclosed within double quotes

Values should be enclosed in double quotation marks to preserve the case and to avoid the upper casing of values.

If the value contains characters enclosed in double quotation marks, then it must be enclosed in single quotation marks.

For example, password = "scoTT" where the value is "scoTT".

3. Values containing characters enclosed in both single and double quotes

If the value contains characters enclosed in both single and double quotation marks, the quotation mark used to enclose the value must be doubled each time it occurs within the value.

For example, password = '"sco''TT"' where the value is "sco'TT".

4. Values containing spaces

All leading and trailing spaces are ignored, but the spaces between the value are recognized. If the value needs to have leading or trailing spaces then it must be enclosed in double quotation marks.

For example, User ID = Sco TT where the value is <Sco TT>.

For example, User ID = "Sco TT " where the value is <Sco TT>.

5. Keywords occurring multiple times in a connection string

If a specific keyword occurs multiple times in a connection string, the last occurrence listed is used in the value set.

For example, with "User ID = scott; password = tiger; User ID = david" connection string, User ID value is david.

To limit malicious access, the APIs of OracleConnectionStringBuilder, Add and Item, insert double quotes around connection string attributes that use special characters. Using special characters in strings can be a format string attack method. Adding double quotes for attributes will not affect database access in most cases. However, there are attributes that require exact string matching, such as passwords. OracleConnectionStringBuilder's addition of double quotes to a password with special characters may result in a generation of a connection string that fails to authenticate against the database.

Example

```
// C#

using System;
using System.Data;
using System.Data.Common;
using Oracle.DataAccess.Client;
using System.Collections;

class ConnectionStringBuilderSample
{
    static void Main(string[] args)
    {
        bool bRet = false;

        // Create an instance of OracleConnectionStringBuilder
        OracleConnectionStringBuilder connStrBuilder =
            new OracleConnectionStringBuilder();

        // Add new key/value pairs to the connection string
        connStrBuilder.Add("User Id", "scott");
        connStrBuilder.Add("Password", "tiger");
```

```

connStrBuilder.Add("Data Source", "oracle");
connStrBuilder.Add("pooling", false);

// Modify the existing value
connStrBuilder["Data source"] = "inst1";

// Remove an entry from the connection string
bRet = connStrBuilder.Remove("pooling");

//ContainsKey indicates whether or not the specific key exist
//returns true even if the user has not specified it explicitly
Console.WriteLine("Enlist exist: " +
    connStrBuilder.ContainsKey("Enlist"));

//returns false
connStrBuilder.ContainsKey("Invalid");

// ShouldSerialize indicates whether or not a specific key
// exists in connection string inherited from DbConnectionStringBuilder.
// returns true if the key is explicitly added the user otherwise false;
// this will return false as this key doesn't exists.
connStrBuilder.ShouldSerialize("user");

// returns false because this key is not added by user explicitly.
connStrBuilder.ShouldSerialize("Enlist");

// IsFixedSize [read-only property]
Console.WriteLine("Connection String is fixed size only: "
    + connStrBuilder.IsFixedSize);
Console.WriteLine("Key/Value Pair Count: " + connStrBuilder.Count);

//adding a new key which is not supported by the provider
//is not allowed.
try
{
    //this will throw an exception.
    connStrBuilder.Add("NewKey", "newValue");
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}

Console.WriteLine("Key/Value Pair Count: " + connStrBuilder.Count);

//modifying a existing key is allowed.
connStrBuilder.Add("Enlist", false);
Console.WriteLine("Key/Value Pair Count: " + connStrBuilder.Count);

// Get all the keys and values supported by the provider.
ICollection keyCollection = connStrBuilder.Keys;
ICollection valueCollection = connStrBuilder.Values;

IEnumerator keys = keyCollection.GetEnumerator();
IEnumerator values = valueCollection.GetEnumerator();

while (keys.MoveNext())
{
    values.MoveNext();
    Console.WriteLine("Key: {0}    Value: {1} \n"
        , keys.Current , values.Current);
}

```



```
}  
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Members](#)
- [OracleConnectionStringBuilder Constructors](#)
- [OracleConnectionStringBuilder Public Properties](#)
- [OracleConnectionStringBuilder Public Methods](#)

OracleConnectionStringBuilder Members

OracleConnectionStringBuilder members are listed in the following tables.

OracleConnectionStringBuilder Constructors

OracleConnectionStringBuilder constructors are listed in [Table 6-57](#).

Table 6-57 OracleConnectionStringBuilder Constructors

Constructor	Description
OracleConnectionStringBuilder Constructors	Instantiates a new instance of OracleConnectionStringBuilder class (Overloaded)

OracleConnectionStringBuilder Public Properties

OracleConnectionStringBuilder instance properties are listed in [Table 6-58](#).

Table 6-58 OracleConnectionStringBuilder Public Properties

Properties	Description
BrowsableConnectionString	Inherited from System.Data.Common.DbConnectionStringBuilder
ConnectionLifeTime	Specifies the value corresponding to the Connection Lifetime attribute in the ConnectionString property
ConnectionString	Inherited from System.Data.Common.DbConnectionStringBuilder
ConnectionTimeout	Specifies the value corresponding to the Connection Timeout attribute in the ConnectionString property
ContextConnection	Specifies the value corresponding to the Context Connection attribute in the ConnectionString property
Count	Inherited from System.Data.Common.DbConnectionStringBuilder
DataSource	Specifies the value corresponding to the Data Source attribute in the ConnectionString property

Table 6-58 (Cont.) OracleConnectionStringBuilder Public Properties

Properties	Description
DBAPrivilege	Specifies the value corresponding to the DBA Privilege attribute in the ConnectionString property
DecrPoolSize	Specifies the value corresponding to the Decr Pool Size attribute in the ConnectionString property
Enlist	Specifies the value corresponding to the Enlist attribute in the ConnectionString property
HAEvents	Specifies the value corresponding to the HA Events attribute in the ConnectionString property
IncrPoolSize	Specifies the value corresponding to the Incr Pool Size attribute in the ConnectionString property
IsFixedSize	Indicates whether or not the Connection String Builder has a fixed size
IsReadOnly	Inherited from System.Data.Common.DbConnectionStringBuilder
Item	Specifies the value associated with the specified attribute
Keys	Specifies a collection of attributes contained in the Connection String Builder
LoadBalancing	Specifies the value corresponding to the Load Balancing attribute in the ConnectionString property
MaxPoolSize	Specifies the value corresponding to the Max Pool Size attribute in the ConnectionString property
MetadataPooling	Specifies the value that corresponds to the Metadata Pooling attribute in the ConnectionString property
MinPoolSize	Specifies the value corresponding to the Min Pool Size attribute in the ConnectionString property
Password	Specifies the value corresponding to the Password attribute in the ConnectionString property
PersistSecurityInfo	Specifies the value corresponding to the Persist Security Info attribute in the ConnectionString property
Pooling	Specifies the value corresponding to the Pooling attribute in the ConnectionString property
ProxyPassword	Specifies the value corresponding to the Proxy User Id attribute in the ConnectionString property
ProxyUserId	Specifies the value corresponding to the Proxy User Id attribute in the ConnectionString property
SelfTuning	Specifies the value corresponding to the Self Tuning attribute in the ConnectionString property
StatementCachePurge	Specifies the value corresponding to the Statement Cache Purge attribute in the ConnectionString property
StatementCacheSize	Specifies the value corresponding to the Statement Cache Size attribute in the ConnectionString property
TnsAdmin	Specifies the directory where ODP.NET can find its sqlnet.ora and tnsnames.ora configuration files

Table 6-58 (Cont.) OracleConnectionStringBuilder Public Properties

Properties	Description
TokenAuthentication	Specifies the value corresponding to the token_auth attribute in the ConnectionString property. <i>This property is supported by managed ODP.NET and ODP.NET Core only.</i>
TokenLocation	Specifies the value corresponding to the token_location attribute in the ConnectionString property. <i>This property is supported by managed ODP.NET and ODP.NET Core only.</i>
UserID	Specifies the value corresponding to the User Id attribute in the ConnectionString property
ValidateConnection	Specifies the value corresponding to the Validate Connection attribute in the ConnectionString property
Values	Specifies a collection of values contained in the Connection String Builder
WalletLocation	Specifies the ODP.NET wallet directory
WalletPassword	Stores the password to decrypt the PKCS #12 wallet <i>This property is supported by managed ODP.NET and ODP.NET Core only.</i>

OracleConnectionStringBuilder Public Methods

OracleConnectionStringBuilder instance methods are listed in [Table 6-59](#).

Table 6-59 OracleConnectionStringBuilder Public Methods

Methods	Description
Add	Inherited from System.Data.Common.DbConnectionStringBuilder
Clear	Clears the connection string contents
ContainsKey	Indicates whether or not a specific attribute in the connection string is supported by ODP.NET
EquivalentTo	Inherited from System.Data.Common.DbConnectionStringBuilder
Remove	Removes the entry corresponding to the specified attribute from the connection string
ShouldSerialize	Inherited from System.Data.Common.DbConnectionStringBuilder
ToString	Inherited from System.Data.Common.DbConnectionStringBuilder
TryGetValue	Returns the value corresponding to the supplied attribute, as an output parameter

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)

OracleConnectionStringBuilder Constructors

OracleConnectionStringBuilder constructors instantiate new instances of the OracleConnectionStringBuilder class.

Overload List:

- [OracleConnectionStringBuilder\(\)](#)
This constructor instantiates a new instance of OracleConnectionStringBuilder class.
- [OracleConnectionStringBuilder\(string\)](#)
This constructor instantiates a new instance of the OracleConnectionStringBuilder class with the provided connection string.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

OracleConnectionStringBuilder()

This constructor instantiates a new instance of the OracleConnectionStringBuilder class.

Declaration

```
// C#  
public OracleConnectionStringBuilder();
```

Remarks

The ConnectionString property is empty after the object is created.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)

OracleConnectionStringBuilder(string)

This constructor instantiates a new instance of the OracleConnectionStringBuilder class with the provided connection string.

Declaration

```
// C#
public OracleConnectionStringBuilder(string connectionString);
```

Parameters

- *connectionString*
The connection information.

Exceptions

ArgumentNullException - The *connectionString* parameter is null.

ArgumentException - The *connectionString* parameter is invalid.

Remarks

The *ConnectionString* property of this instance is set to the supplied connection string.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

OracleConnectionStringBuilder Public Properties

OracleConnectionStringBuilder public properties are listed in [Table 6-60](#).

Table 6-60 OracleConnectionStringBuilder Public Properties

Properties	Description
BrowsableConnectionString	Inherited from System.Data.Common.DbConnectionStringBuilder
ConnectionLifetime	Specifies the value corresponding to the Connection Lifetime attribute in the ConnectionString property
ConnectionString	Inherited from System.Data.Common.DbConnectionStringBuilder
ConnectionTimeout	Specifies the value corresponding to the Connection Timeout attribute in the ConnectionString property
ContextConnection	Specifies the value corresponding to the Context Connection attribute in the ConnectionString property
Count	Inherited from System.Data.Common.DbConnectionStringBuilder
DataSource	Specifies the value corresponding to the Data Source attribute in the ConnectionString property

Table 6-60 (Cont.) OracleConnectionStringBuilder Public Properties

Properties	Description
DBAPrivilege	Specifies the value corresponding to the DBA Privilege attribute in the ConnectionString property
DecrPoolSize	Specifies the value corresponding to the Decr Pool Size attribute in the ConnectionString property
Enlist	Specifies the value corresponding to the Enlist attribute in the ConnectionString property
HAEvents	Specifies the value corresponding to the HA Events attribute in the ConnectionString property
IncrPoolSize	Specifies the value corresponding to the Incr Pool Size attribute in the ConnectionString property
IsFixedSize	Indicates whether or not the Connection String Builder has a fixed size
IsReadOnly	Inherited from System.Data.Common.DbConnectionStringBuilder
Item	Specifies the value associated with the specified attribute
Keys	Specifies a collection of attributes contained in the Connection String Builder
LoadBalancing	Specifies the value corresponding to the Load Balancing attribute in the ConnectionString property
MaxPoolSize	Specifies the value corresponding to the Max Pool Size attribute in the ConnectionString property
MetadataPooling	Specifies the value that corresponds to the Metadata Pooling attribute in the ConnectionString property
MinPoolSize	Specifies the value corresponding to the Min Pool Size attribute in the ConnectionString property
Password	Specifies the value corresponding to the Password attribute in the ConnectionString property
PersistSecurityInfo	Specifies the value corresponding to the Persist Security Info attribute in the ConnectionString property
Pooling	Specifies the value corresponding to the Pooling attribute in the ConnectionString property
ProxyPassword	Specifies the value corresponding to the Proxy User Id attribute in the ConnectionString property
ProxyUserId	Specifies the value corresponding to the Proxy User Id attribute in the ConnectionString property
SelfTuning	Specifies the value corresponding to the Self Tuning attribute in the ConnectionString property
StatementCachePurge	Specifies the value corresponding to the Statement Cache Purge attribute in the ConnectionString property
StatementCacheSize	Specifies the value corresponding to the Statement Cache Size attribute in the ConnectionString property
TnsAdmin	Specifies the directory where ODP.NET can find its sqlnet.ora and tnsnames.ora configuration files

Table 6-60 (Cont.) OracleConnectionStringBuilder Public Properties

Properties	Description
TokenAuthentication	Specifies the value corresponding to the token_auth attribute in the ConnectionString property. This property is supported by managed ODP.NET and ODP.NET Core only.
TokenLocation	Specifies the value corresponding to the token_location attribute in the ConnectionString property. This property is supported by managed ODP.NET and ODP.NET Core only.
UserID	Specifies the value corresponding to the User Id attribute in the ConnectionString property
ValidateConnection	Specifies the value corresponding to the Validate Connection attribute in the ConnectionString property
Values	Specifies a collection of values contained in the Connection String Builder
WalletLocation	Specifies the ODP.NET wallet directory
WalletPassword	Stores the password to decrypt the PKCS #12 wallet <i>This property is supported by managed ODP.NET and ODP.NET Core only.</i>

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

ConnectionLifeTime

This property specifies the value corresponding to the Connection LifeTime attribute in the ConnectionString property.

Declaration

```
// C#  
public int ConnectionLifeTime{get; set;}
```

Property Value

An int that represents the value of the supplied attribute.

Exceptions

OracleException - The specified value is less than zero.

Remarks

When an `OracleConnectionStringBuilder` instance is created, this property is set to the default value of the corresponding connection string attribute.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

ConnectionTimeout

This property specifies the value corresponding to the Connection Timeout attribute in the `ConnectionString` property.

Declaration

```
// C#  
public int ConnectionTimeout{get; set;}
```

Property Value

An `int` that represents the value of the supplied attribute.

Exceptions

`OracleException` - The specified value is less than zero.

Remarks

When an `OracleConnectionStringBuilder` instance is created, this property is set to the default value of the corresponding connection string attribute.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

ContextConnection

This property specifies the value corresponding to the Context Connection attribute in the `ConnectionString` property.

Declaration

```
// C#  
public bool ContextConnection {get; set;}
```

Property Value

A bool that represents the value of the supplied attribute.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

DataSource

This property specifies the value corresponding to the Data Source attribute in the ConnectionString property.

Declaration

```
// C#  
public string DataSource{get; set;}
```

Property Value

A string that represents the value of the supplied attribute.

Exceptions

ArgumentNullException - The specified value is null.

Remarks

When an OracleConnectionStringBuilder instance is created, this property is set to the default value of the corresponding connection string attribute.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

DBAPrivilege

This property specifies the value corresponding to the DBA Privilege attribute in the ConnectionString property.

Declaration

```
// C#  
public string DBAPrivilege{get; set;}
```

Property Value

A string that represents the value of the supplied attribute.

Possible values are SYSDBA, SYSASM, SYSOPER, SYSBACKUP, SYSDG, SYSKM, or SYSRAC.

Exceptions

ArgumentNullException - The specified value is null.

OracleException - The specified value is invalid.

Remarks

When an OracleConnectionStringBuilder instance is created, this property is set to the default value of the corresponding connection string attribute.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

DecrPoolSize

This property specifies the value corresponding to the Decr Pool Size attribute in the ConnectionString property.

Declaration

```
// C#  
public int DecrPoolSize{get; set;}
```

Property Value

An int that represents the value of the supplied attribute.

Exceptions

OracleException - The specified value is less than 1.

Remarks

When an OracleConnectionStringBuilder instance is created, this property is set to the default value of the corresponding connection string attribute.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

Enlist

This property specifies the value corresponding to the Enlist attribute in the `ConnectionString` property.

Declaration

```
// C#  
public string Enlist{get; set;};
```

Property Value

A string that represents the value of the supplied attribute. Values are case-insensitive. Possible values are: `dynamic`, `true`, `false`, `yes`, and `no`.

Exceptions

`ArgumentNullException` - The specified value is null.

`OracleException` - The supplied value is not one of following: `dynamic`, `true`, `false`, `yes`, or `no`.

Remarks

When an `OracleConnectionStringBuilder` instance is created, this property is set to the default value of the corresponding connection string attribute.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

HAEvents

This property specifies the value corresponding to the HA Events attribute in the `ConnectionString` property.

Declaration

```
// C#  
public bool HAEvents{get; set;}
```

Property Value

A `bool` that represents the value of the supplied attribute.

Remarks

When an `OracleConnectionStringBuilder` instance is created, this property is set to the default value of the corresponding connection string attribute.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

IncrPoolSize

This property specifies the value corresponding to the `Incr Pool Size` attribute in the `ConnectionString` property.

Declaration

```
// C#  
public int IncrPoolSize{get; set;}
```

Property Value

An `int` that represents the value of the supplied attribute.

Exceptions

`OracleException` - The specified value is less than 1.

Remarks

When an `OracleConnectionStringBuilder` instance is created, this property is set to the default value of the corresponding connection string attribute.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

IsFixedSize

Indicates whether or not the `Connection String Builder` has a fixed size.

Declaration

```
// C#  
public override bool IsFixedSize{get;}
```

Property Value

Returns true if the Connection String Builder has a fixed size; otherwise, returns false.

Remarks

Attributes cannot be added or removed. They can only be modified for connection strings with a fixed size.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

Item

This property specifies the value associated with the specified attribute.

Declaration

```
// C#  
public override object this[string keyword]{get; set;}
```

Property Value

An object value corresponding to the attribute.

Exceptions

ArgumentNullException - The specified attribute is null.

OracleException - The specified attribute is not supported or the specified value is invalid.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

Keys

This property specifies a collection of attributes contained in the Connection String Builder.

Declaration

```
// C#  
public override ICollection Keys{get;}
```

Property Value

Returns an ICollection that represents the attributes in the Connection String Builder.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

LoadBalancing

This property specifies the value corresponding to the Load Balancing attribute in the ConnectionString property.

Declaration

```
// C#  
public bool LoadBalancing {get; set;}
```

Property Value

A bool that contains the value of the supplied attribute.

Remarks

When an OracleConnectionStringBuilder instance is created, this property is set to the default value of the corresponding connection string attribute.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

MaxPoolSize

This property specifies the value corresponding to the Max Pool Size attribute in the ConnectionString property.

Declaration

```
// C#  
public int MaxPoolSize{get; set;}
```

Property Value

An int that represents the value of the supplied attribute.

Exceptions

OracleException - The specified value is less than 1.

Remarks

When an OracleConnectionStringBuilder instance is created, this property is set to the default value of the corresponding connection string attribute.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

MetadataPooling

This property specifies the value that corresponds to the Metadata Pooling attribute in the ConnectionString property.

Declaration

```
// C#  
public bool MetadataPooling{get; set;};
```

Property Value

A bool containing the value of the supplied attribute.

Remarks

When an OracleConnectionStringBuilder instance is created, this property is set to the default value of the corresponding connection string attribute.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

MinPoolSize

This property specifies the value corresponding to the Min Pool Size attribute in the ConnectionString property.

Declaration

```
// C#  
public int MinPoolSize{get; set;}
```

Property Value

An int that contains the value of the supplied attribute.

Exceptions

OracleException - The specified value is less than 0.

Remarks

When an OracleConnectionStringBuilder instance is created, this property is set to the default value of the corresponding connection string attribute.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

Password

This property specifies the value corresponding to the Password attribute in the ConnectionString property.

Declaration

```
// C#  
public string Password{get; set;}
```

Property Value

A string that contains the value of the supplied attribute.

Exception

ArgumentNullException - The specified value is null.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

PersistSecurityInfo

This property specifies the value corresponding to the Persist Security Info attribute in the ConnectionString property.

Declaration

```
// C#  
public bool PersistSecurityInfo{get; set;}
```

Property Value

A bool that represents the value of the supplied attribute.

Remarks

When an OracleConnectionStringBuilder instance is created, this property gets set to the default value of the corresponding connection string attribute.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

Pooling

This property specifies the value corresponding to the Pooling attribute in the ConnectionString property.

Declaration

```
// C#  
public bool Pooling {get; set;}
```

Property Value

A bool that represents the value of the supplied attribute.

Remarks

When an OracleConnectionStringBuilder instance is created, this property is set to the default value of the corresponding connection string attribute.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

ProxyPassword

This property specifies the value corresponding to the Proxy Password attribute in the ConnectionString property.

Declaration

```
// C#  
public string ProxyPassword {get; set;}
```

Property Value

A string that represents the value of the supplied attribute.

Exception

ArgumentNullException - The specified value is null.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

ProxyUserId

This property specifies the value corresponding to the Proxy User Id attribute in the ConnectionString property.

Declaration

```
// C#  
public string ProxyUserId {get; set;}
```

Property Value

A string that represents the value of the supplied attribute.

Exception

ArgumentNullException - The specified value is null.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

SelfTuning

This property specifies the value corresponding to the Self Tuning attribute in the ConnectionString property.

Declaration

```
// C#  
public bool SelfTuning {get; set;}
```

Property Value

A bool that represents the value of the supplied attribute.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

StatementCachePurge

This property specifies the value corresponding to the Statement Cache Purge attribute in the ConnectionString property.

Declaration

```
// C#  
public bool StatementCachePurge {get; set;}
```

Property Value

A bool that represents the value of the supplied attribute.

Remarks

When an OracleConnectionStringBuilder instance is created, this property is set to the default value of the corresponding connection string attribute.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

StatementCacheSize

This property specifies the value corresponding to the Statement Cache Size attribute in the ConnectionString property.

Declaration

```
// C#  
public int StatementCacheSize{get; set;}
```

Property Value

An int that represents the value of the supplied attribute.

Exceptions

OracleException - The specified value is less than zero.

Remarks

When an OracleConnectionStringBuilder instance is created, this property is set to the default value of the corresponding connection string attribute.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

TnsAdmin

This property specifies the directory where ODP.NET can find its sqlnet.ora and tnsnames.ora configuration files.

Declaration

```
// C#  
public string TnsAdmin{get; set;}
```

Property Value

A string that represents the value of the supplied attribute.

Remarks

The default value is an empty string.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

TokenAuthentication

This property specifies the value corresponding to the token_auth attribute in the ConnectionString.

Declaration

```
// C#  
public OracleTokenAuth TokenAuthentication { get; set; }
```

Property Value

An OracleTokenAuth value.

Exceptions

None.

Remarks

When an OracleConnectionStringBuilder instance is created, this property is set to the default value of the corresponding connection string attribute.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

TokenLocation

This property specifies the value corresponding to the token_location attribute in the ConnectionString property.

Declaration

```
// C#  
public string TokenLocation { get; set; }
```

Property Value

A string that represents the value of the supplied attribute.

Exceptions

None.

Remarks

When an OracleConnectionStringBuilder instance is created, this property is set to the default value of the corresponding connection string attribute.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

UserID

This property specifies the value corresponding to the User Id attribute in the `ConnectionString` property.

Declaration

```
// C#  
public string UserID{get; set;}
```

Property Value

A string that represents the value of the supplied attribute.

Exception

`ArgumentNullException` - The specified value is null.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

ValidateConnection

This property specifies the value corresponding to the Validate Connection attribute in the `ConnectionString` property.

Declaration

```
// C#  
public bool ValidateConnection{get; set;}
```

Property Value

A `bool` that represents the value of the supplied attribute.

Remarks

When an `OracleConnectionStringBuilder` instance is created, this property is set to the default value of the corresponding connection string attribute.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

Values

This property specifies a collection of values contained in the Connection String Builder.

Declaration

```
// C#  
public override ICollection Values{get;}
```

Property Value

Returns an *ICollection* that represents the values in the Connection String Builder.

Remarks

The order of the values in the *ICollection* is unspecified, but is the same as the associated attributes in the *ICollection* returned by the *Keys* property.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

WalletLocation

This property specifies the ODP.NET wallet directory.

Declaration

```
// C#  
public string WalletLocation{get; set;}
```

Property Value

A string that represents the value of the supplied attribute.

Remarks

The default value is an empty string.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

WalletPassword

This property stores the password to decrypt the PKCS #12 wallet.

Declaration

```
// C#  
public string WalletPassword {get; set;}
```

Property Value

The connection string attribute used to decrypt the PKCS #12 wallet.

Remarks

Wallet passwords support the same functionality as traditional ODP.NET passwords.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

OracleConnectionStringBuilder Public Methods

OracleConnectionStringBuilder public methods are listed in [Table 6-61](#).

Table 6-61 OracleConnectionStringBuilder Public Methods

Methods	Description
Add	Inherited from System.Data.Common.DbConnectionStringBuilder
Clear	Clears the connection string contents
ContainsKey	Indicates whether or not a specific attribute in the connection string is supported by ODP.NET
EquivalentTo	Inherited from System.Data.Common.DbConnectionStringBuilder
Remove	Removes the entry corresponding to the specified attribute from the connection string

Table 6-61 (Cont.) OracleConnectionStringBuilder Public Methods

Methods	Description
ShouldSerialize	Inherited from System.Data.Common.DbConnectionStringBuilder
ToString	Inherited from System.Data.Common.DbConnectionStringBuilder
TryGetValue	Returns the value corresponding to the supplied attribute, as an output parameter

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

Clear

This method clears the connection string contents.

Declaration

```
// C#  
public override void Clear();
```

Remarks

All key/value pairs are removed from the OracleConnectionStringBuilder object and the ConnectionString property is set to Empty.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

ContainsKey

This method indicates whether or not a specific attribute in the connection string is supported by ODP.NET.

Declaration

```
// C#  
public override bool ContainsKey(string keyword);
```

Parameters

- *keyword*
The attribute being verified.

Return Value

Returns true if the specified attribute exists; otherwise, returns false.

Exceptions

ArgumentNullException - The specified attribute is null.

Remarks

This method indicates if the attribute is part of the provider-supported attributes. It does not indicate if the user added the attribute to the connection string.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

Remove

This method removes the entry corresponding to the specified attribute from the connection string.

Declaration

```
// C#  
public override bool Remove(string keyword);
```

Parameters

- *keyword*
The attribute that specifies the entry to be removed.

Return Value

Returns true if the attribute existed in the connection string and the corresponding entry was removed; otherwise, returns false.

Exceptions

ArgumentNullException - The specified attribute is null.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

TryGetValue

This method returns the value corresponding to the supplied attribute, as an output parameter.

Declaration

```
// C#  
public override bool TryGetValue(string keyword, out object value);
```

Parameters

- *keyword*
The attribute for which the value is being retrieved.
- *value*
The value of the supplied attribute.
Sets *value* to the default value if the attribute is not present in the connection string.

Return Value

Returns `true` if the value that corresponds to the attribute has been successfully retrieved; otherwise, returns `false`. If the attribute is not present in the connection string, returns `false` and sets the *value* to null.

Exceptions

`ArgumentNullException` - The specified attribute is null.

Remarks

If the function returns `false`, sets *value* to null.

If the attribute is not present in the connection string, sets *value* to the default value.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleConnectionStringBuilder Class](#)
- [OracleConnectionStringBuilder Members](#)

OracleCredential Class

OracleCredential class provides a secure way to provide password while opening connection with Oracle Database using the ODP.NET driver. Use this class to avoid providing passwords in clear text in the connection string while opening a connection with Oracle Database. One can pass user id, password and DBA Privilege specific attributes through OracleCredential constructors and thus these attributes do not need to be in the connection string.

Operating system authenticated and context connections are not supported through OracleCredential class. An ArgumentException will be thrown if "/" is passed for userId or proxyUserId arguments of OracleCredential constructor.

ODP.NET uses a modified connection pooling algorithm with OracleCredential, which is described in more detail in the section [Connection Pooling with OracleCredential](#).

Class Inheritance

System.Object
Oracle.DataAccess.Client.OracleCredential

Declaration

```
// C#  
public sealed class OracleCredential
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#  
using System;  
using System.Data;  
using System.Security;  
using Oracle.DataAccess.Client;  
//using Oracle.ManagedDataAccess.Client;  
  
class OracleCredentialSample
```

```
{
    static void Main()
    {
        // Connect
        string constr = "Data Source=oracle";

        SecureString secPwd = new SecureString();
        secPwd.AppendChar('h');
        secPwd.AppendChar('r');

        // Make the password read-only.
        secPwd.MakeReadOnly();

        // Create OracleCredential with userid and secure password.
        OracleCredential oc = new OracleCredential("hr", secPwd);

        OracleConnection con = new OracleConnection(constr, oc);
        con.Open();

        // Execute a SQL SELECT

        OracleCommand cmd = con.CreateCommand();
        cmd.CommandText = "select * from employees";

        OracleDataReader reader = cmd.ExecuteReader();

        // Print all employee numbers
        while (reader.Read())
            Console.WriteLine(reader.GetInt32(0));

        // Clean up
        reader.Dispose();
        cmd.Dispose();
        con.Dispose();
    }
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCredential Members](#)
- [OracleCredential Constructors](#)
- [OracleCredential Properties](#)
- [OracleCredential Public Methods](#)
- [Connection Pooling with OracleCredential](#)

OracleCredential Members

OracleCredential members are listed in the following tables.

OracleCredential Constructors

OracleCredential constructors are listed in [Table 6-62](#).

Table 6-62 OracleCredential Constructors

Constructor	Description
OracleCredential Constructors	Instantiates a new instance of OracleCredential class (Overloaded)

OracleCredential Properties

OracleCredential properties are listed in [Table 6-63](#).

Table 6-63 OracleCredential Properties

Property	Description
DBAPrivilege	Returns the DBA Privilege used to create the OracleCredential object. OracleDBAPrivilege.None is returned if it was not provided by the user.
Password	Returns the password of the OracleCredential object
ProxyPassword	Returns the proxy user password of the OracleCredential object
ProxyUserId	Returns the proxy user id of the OracleCredential object
UserId	Returns the user id of the OracleCredential object

OracleCredential Public Methods

OracleCredential public methods are listed in [Table 6-70](#).

Table 6-64 OracleCredential Public Methods

Public Method	Description
Equals(Object)	Inherited from System.Object
Equals(Object)	Inherited from System.Object
GetHashCode()	Inherited from System.Object
GetType()	Inherited from System.Object
SetPassword	Sets or updates the password on the OracleCredential instance <i>Not available in ODP.NET, Unmanaged Driver</i>
SetProxyPassword	Sets or updates the proxy password on the OracleCredential instance <i>Not available in ODP.NET, Unmanaged Driver</i>
ToString()	Inherited from System.Object

OracleCredential Constructors

OracleCredential constructors instantiate new instances of an OracleCredential class.

Overload List:

- [OracleCredential\(string, SecureString\)](#)

This constructor creates an `OracleCredential` object with the user id and the secured password.

- [OracleCredential\(string, SecureString, OracleDBAPrivilege\)](#)

This constructor creates an `OracleCredential` object with the user id, the secured password, and the DBA privilege request.

- [OracleCredential\(string, SecureString, string, SecureString\)](#)

This constructor creates an `OracleCredential` object with the user id and the secured password for the end user, as well as the user id and the secured password for the proxy user.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCredential Class](#)
- [OracleCredential Members](#)

OracleCredential(string, SecureString)

This constructor creates an `OracleCredential` object with the user id and the secured password.

Declaration

```
// C#  
public OracleCredential(string userId, SecureString password)
```

Exceptions

- `ArgumentNullException` is raised if any parameters are passed as null.
- `ArgumentException` is raised if "/" is passed for `userId` argument.
- `ArgumentException` is raised if the `SecureString` containing the password is not read-only.
- `InvalidOperationException` is raised when `OracleCredential` object is used with any of user id, password, proxy user id, proxy password and/or dba privilegeconnection string attribute(s) present in the connection string.

Remarks

With this constructor, `OracleDBAPrivilege.None` is used for the DBA privilege when requesting for a connection.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCredential Class](#)
- [OracleCredential Members](#)

OracleCredential(string, SecureString, OracleDBAPrivilege)

This constructor creates an `OracleCredential` object with the user id, the secured password, and the DBA privilege request.

Declaration

```
// C#  
public OracleCredential(string userId, SecureString password, OracleDBAPrivilege  
dbaprivilege)
```

Exceptions

- `ArgumentNullException` is raised if any parameters are passed as null.
- `ArgumentException` is raised if "/" is passed for `userId` argument.
- `ArgumentException` is raised if the `SecureString` containing the password is not read-only.
- `InvalidOperationException` is raised when `OracleCredential` object is used with any of user id, password, proxy user id, proxy password and/or dba privilegeconnection string attribute(s) present in the connection string.

Remarks

With this constructor, the specified `dbaprivilege` is for the DBA privilege when requesting for a connection.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)
- [OracleDBAPrivilege Enumeration](#)

OracleCredential(string, SecureString, string, SecureString)

This constructor creates an `OracleCredential` object with the user id and the secured password for the end user, as well as the user id and the secured password for the proxy user.

Declaration

```
// C#  
public OracleCredential(string userId, SecureString password, string proxyUserId,  
SecureString proxyPassword)
```

Exceptions

- `ArgumentNullException` is raised if any parameters are passed as null.
- `ArgumentException` is raised if "/" is passed for `userId` argument.
- `ArgumentException` is raised if the `SecureString` containing the password is not read-only.

- InvalidOperationException is raised when OracleCredential object is used with any of user id, password, proxy user id, proxy password and/or dba privilegeconnection string attribute(s) present in the connection string.

Remarks

With this constructor, OracleDBAPrivilege.None is for the DBA privilege when requesting for a connection.

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCredential Class](#)
- [OracleCredential Members](#)

OracleCredential Properties

OracleCredential properties are listed in [Table 6-65](#).

Table 6-65 OracleCredential Properties

Property	Description
DBAPrivilege	Returns the DBA Privilege used to create the OracleCredential object. OracleDBAPrivilege.None is returned if it was not provided by the user.
Password	Returns the password of the OracleCredential object.
ProxyPassword	Returns the proxy user password of the OracleCredential object.
ProxyUserId	Returns the proxy user id of the OracleCredential object.
UserId	Returns the user id of the OracleCredential object.

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCredential Class](#)
- [OracleCredential Members](#)

DBAPrivilege

This property returns the DBA Privilege that is associated with the user.

Declaration

```
// C#
OracleDBAPrivilege DBAPrivilege { get; }
```

Property Value

The `OracleDBAPrivilege` enum value that is associated with the user.

Remarks

`OracleDBAPrivilege.None` is returned if it was not set by the user.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCredential Class](#)
- [OracleCredential Members](#)
- [OracleDBAPrivilege Enumeration](#)

Password

This property returns the encrypted password of the user.

Declaration

```
// C#  
SecureString Password { get; }
```

Property Value

The encrypted password of the user.

`OracleCredential` does not support double quotes around a `SecureString` password. Double quotes can be used within a password, however.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCredential Class](#)
- [OracleCredential Members](#)

ProxyPassword

This property returns the encrypted password of the proxy user.

Declaration

```
// C#  
SecureString ProxyUserPassword { get; }
```

Property Value

The encrypted password of the proxy user.

Remarks

OracleCredential does not support double quotes around a SecureString password. Double quotes can be used within a password, however.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleCredential Class](#)
- [OracleCredential Members](#)

ProxyUserId

This property returns the proxy user id.

Declaration

```
// C#  
string ProxyUserId { get; }
```

Property Value

The proxy user id.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleCredential Class](#)
- [OracleCredential Members](#)

UserId

This property returns the user id.

Declaration

```
// C#  
string UserId { get; }
```

Property Value

The user id.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCredential Class](#)
- [OracleCredential Members](#)

OracleCredential Public Methods

OracleCredential public methods are listed in [Table 6-66](#).

Table 6-66 OracleCredential Public Methods

Property	Description
Equals(Object)	Inherited from System.Object
GetHashCode()	Inherited from System.Object
GetType()	Inherited from System.Object
SetPassword	Sets or updates the password on the OracleCredential instance <i>Not available in ODP.NET, Unmanaged Driver</i>
SetProxyPassword	Sets or updates the proxy password on the OracleCredential instance <i>Not available in ODP.NET, Unmanaged Driver</i>
ToString()	Inherited from System.Object

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCredential Class](#)
- [OracleCredential Members](#)

SetPassword

This method sets or updates the password on the OracleCredential instance.

Declaration

```
// C#
public void SetPassword(OracleOpaqueString opaquePassword)
```

Parameters

- *opaquePassword*
new user password

Exceptions

Throws ORA-50129 if user id is not set on the `OracleCredential`.

Throws `ArgumentNullException` if *opaquePassword* is null.

Throws `ArgumentException` if password has length of 0.

Throws ORA-50128 if *opaquePassword* is not read only.

Remarks

This method cannot update the database user's password.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCredential Class](#)
- [OracleCredential Members](#)

SetProxyPassword

This method sets or updates the proxy password on the `OracleCredential` instance.

Declaration

```
// C#  
public void SetProxyPassword(OracleOpaqueString opaqueProxyPassword)
```

Parameters

- *opaqueProxyPassword*
new proxy password

Exceptions

Throws ORA-50129 if user id is not set on the `OracleCredential`.

Throws `ArgumentNullException` if *opaqueProxyPassword* is null.

Throws `ArgumentException` if password has length of 0.

Throws ORA-50128 if *opaqueProxyPassword* is not read only.

Remarks

This method cannot update the proxy database user's password.

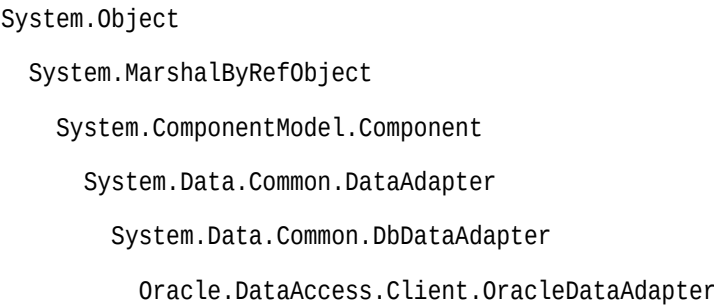
① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCredential Class](#)
- [OracleCredential Members](#)

OracleDataAdapter Class

An `OracleDataAdapter` object represents a data provider object that populates the `DataSet` and updates changes in the `DataSet` to the Oracle database.

Class Inheritance



Declaration

```
// C#
public sealed class OracleDataAdapter : DbDataAdapter, IDbDataAdapter
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

The following example uses the `OracleDataAdapter` and the dataset to update the EMP table:

```
// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;

class OracleDataAdapterSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        string cmdstr = "SELECT empno, sal from emp";

        // Create the adapter with the selectCommand txt and the
        // connection string
        OracleDataAdapter adapter = new OracleDataAdapter(cmdstr, constr);

        // Create the builder for the adapter to automatically generate
        // the Command when needed
        OracleCommandBuilder builder = new OracleCommandBuilder(adapter);

        // Create and fill the DataSet using the EMP
        DataSet dataset = new DataSet();
        adapter.Fill(dataset, "EMP");

        // Get the EMP table from the dataset
        DataTable table = dataset.Tables["EMP"];

        // Indicate DataColumn EMPNO is unique
        // This is required by the OracleCommandBuilder to update the EMP table
        table.Columns["EMPNO"].Unique = true;

        // Get the first row from the EMP table
        DataRow row = table.Rows[0];

        // Update the salary
        double sal = double.Parse(row["SAL"].ToString());
        row["SAL"] = sal + .01;

        // Now update the EMP using the adapter
        // The OracleCommandBuilder will create the UpdateCommand for the
        // adapter to update the EMP table
        adapter.Update(dataset, "EMP");

        Console.WriteLine("Row updated successfully");
    }
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataAdapter Members](#)
- [OracleDataAdapter Constructors](#)
- [OracleDataAdapter Static Methods](#)
- [OracleDataAdapter Properties](#)
- [OracleDataAdapter Public Methods](#)
- [OracleDataAdapter Events](#)

OracleDataAdapter Members

OracleDataAdapter members are listed in the following tables.

OracleDataAdapter Constructors

OracleDataAdapter constructors are listed in [Table 6-67](#).

Table 6-67 OracleDataAdapter Constructors

Constructor	Description
OracleDataAdapter Constructors	Instantiates a new instance of OracleDataAdapter class (Overloaded)

OracleDataAdapter Static Methods

The OracleDataAdapter static method is listed in [Table 6-68](#).

Table 6-68 OracleDataAdapter Static Method

Method	Description
Equals	Inherited from System.Object (Overloaded)

OracleDataAdapter Properties

OracleDataAdapter properties are listed in [Table 6-69](#).

Table 6-69 OracleDataAdapter Properties

Property	Description
AcceptChangesDuringFill	Inherited from System.Data.Common.DataAdapter
Container	Inherited from System.ComponentModel.Component
ContinueUpdateOnError	Inherited from System.Data.Common.DataAdapter
DeleteCommand	A SQL statement or stored procedure to delete rows from an Oracle database

Table 6-69 (Cont.) OracleDataAdapter Properties

Property	Description
IdentityInsert	Determines whether or not to insert identity column values in the DataSet into the database when the Update method is invoked. <i>Not Available in the ODP.NET, Managed Driver and ODP.NET Core</i>
IdentityUpdate	Determines whether or not to update identity column values in the DataSet into the database when the Update method is invoked. <i>Not Available in the ODP.NET, Managed Driver and ODP.NET Core</i>
InsertCommand	A SQL statement or stored procedure to insert new rows into an Oracle database
MissingMappingAction	Inherited from System.Data.Common.DataAdapter
MissingSchemaAction	Inherited from System.Data.Common.DataAdapter
Requery	Determines whether or not the SelectCommand is reexecuted on the next call to Fill
ReturnProviderSpecificTypes	Determines if the Fill method returns ODP.NET-specific values or .NET common language specification values
SafeMapping	Creates a mapping between column names in the result set to .NET types, to preserve the data <i>Not Available in the ODP.NET, Managed Driver and ODP.NET Core</i>
SelectCommand	A SQL statement or stored procedure that returns a single or multiple result set
Site	Inherited from System.ComponentModel.Component
SuppressGetDecimalInvalidCastException	Specifies whether to suppress the InvalidCastException and return a rounded-off 28 or 29 precision Oracle NUMBER value that can be represented as a .NET decimal.
TableMappings	Inherited from System.Data.Common.DataAdapter
UpdateBatchSize	Specifies a value that enables or disables batch processing support, and specifies the number of SQL statements that can be executed in a single round-trip to the database
UpdateCommand	A SQL statement or stored procedure to update rows from the DataSet to an Oracle database

OracleDataAdapter Public Methods

OracleDataAdapter public methods are listed in [Table 6-70](#).

Table 6-70 OracleDataAdapter Public Methods

Public Method	Description
CreateObjRef	Inherited from System.MarshalByRefObject
Dispose	Inherited from System.ComponentModel.Component
Equals	Inherited from System.Object (Overloaded)

Table 6-70 (Cont.) OracleDataAdapter Public Methods

Public Method	Description
Fill	Adds or refreshes rows in the DataSet to match the data in the Oracle database (Overloaded)
FillSchema	Inherited from System.Data.Common.DbDataAdapter
GetFillParameters	Inherited from System.Data.Common.DbDataAdapter
GetHashCode	Inherited from System.Object
GetLifetimeService	Inherited from System.MarshalByRefObject
GetType	Inherited from System.Object
InitializeLifetimeService	Inherited from System.MarshalByRefObject
ToString	Inherited from System.Object
Update	Inherited from System.Data.Common.DbDataAdapter

OracleDataAdapter Events

OracleDataAdapter events are listed in [Table 6-71](#).

Table 6-71 OracleDataAdapter Events

Event Name	Description
Disposed	Inherited from System.ComponentModel.Component
FillError	Inherited from System.Data.Common.DbDataAdapter
RowUpdated	This event is raised when row(s) have been updated by the Update() method
RowUpdating	This event is raised when row data are about to be updated to the database

OracleDataAdapter Constructors

OracleDataAdapter constructors create new instances of an OracleDataAdapter class.

Overload List:

- [OracleDataAdapter\(\)](#)
This constructor creates an instance of an OracleDataAdapter class.
- [OracleDataAdapter\(OracleCommand\)](#)
This constructor creates an instance of an OracleDataAdapter class with the provided OracleCommand as the SelectCommand.
- [OracleDataAdapter\(string, OracleConnection\)](#)
This constructor creates an instance of an OracleDataAdapter class with the provided OracleConnection object and the command text for the SelectCommand.
- [OracleDataAdapter\(string, string\)](#)
This constructor creates an instance of an OracleDataAdapter class with the provided connection string and the command text for the SelectCommand.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)

OracleDataAdapter()

This constructor creates an instance of an `OracleDataAdapter` class with no arguments.

Declaration

```
// C#  
public OracleDataAdapter();
```

Remarks

Initial values are set for the following `OracleDataAdapter` properties as indicated:

- `MissingMappingAction = MissingMappingAction.Passthrough`
- `MissingSchemaAction = MissingSchemaAction.Add`

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)

OracleDataAdapter(OracleCommand)

This constructor creates an instance of an `OracleDataAdapter` class with the provided `OracleCommand` as the `SelectCommand`.

Declaration

```
// C#  
public OracleDataAdapter(OracleCommand selectCommand);
```

Parameters

- *selectCommand*
The `OracleCommand` that is to be set as the `SelectCommand` property.

Remarks

Initial values are set for the following `OracleDataAdapter` properties as indicated:

- `MissingMappingAction = MissingMappingAction.Passthrough`
- `MissingSchemaAction = MissingSchemaAction.Add`

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)

OracleDataAdapter(string, OracleConnection)

This constructor creates an instance of an `OracleDataAdapter` class with the provided `OracleConnection` object and the command text for the `SelectCommand`.

Declaration

```
// C#  
public OracleDataAdapter(string selectCommandText, OracleConnection  
    selectConnection);
```

Parameters

- *selectCommandText*
The string that is set as the `CommandText` of the `SelectCommand` property of the `OracleDataAdapter`.
- *selectConnection*
The `OracleConnection` to connect to the Oracle database.

Remarks

The `OracleDataAdapter` opens and closes the connection, if it is not already open. If the connection is open, it must be explicitly closed.

Initial values are set for the following `OracleDataAdapter` properties as indicated:

- `MissingMappingAction` = `MissingMappingAction.Passthrough`
- `MissingSchemaAction` = `MissingSchemaAction.Add`

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)

OracleDataAdapter(string, string)

This constructor creates an instance of an `OracleDataAdapter` class with the provided connection string and the command text for the `SelectCommand`.

Declaration

```
// C#  
public OracleDataAdapter(string selectCommandText, string  
    selectConnectionString);
```

Parameters

- *selectCommandText*
The string that is set as the CommandText of the SelectCommand property of the OracleDataAdapter.
- *selectConnectionString*
The connection string.

Remarks

Initial values are set for the following OracleDataAdapter properties as indicated:

- MissingMappingAction = MissingMappingAction.Passthrough
- MissingSchemaAction = MissingSchemaAction.Add

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)

OracleDataAdapter Static Methods

The OracleDataAdapter static method is listed in [Table 6-72](#).

Table 6-72 OracleDataAdapter Static Method

Method	Description
Equals	Inherited from System.Object (Overloaded)

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)

OracleDataAdapter Properties

OracleDataAdapter properties are listed in [Table 6-73](#).

Table 6-73 OracleDataAdapter Properties

Property	Description
AcceptChangesDuringFill	Inherited from System.Data.Common.DataAdapter
Container	Inherited from System.ComponentModel.Component
ContinueUpdateOnError	Inherited from System.Data.Common.DataAdapter
DeleteCommand	A SQL statement or stored procedure to delete rows from an Oracle database
IdentityInsert	Determines whether or not to insert identity column values in the DataSet into the database when the Update method is invoked. <i>Not Available in the ODP.NET, Managed Driver and ODP.NET Core</i>
IdentityUpdate	Determines whether or not to update identity column values in the DataSet into the database when the Update method is invoked. <i>Not Available in the ODP.NET, Managed Driver and ODP.NET Core</i>
InsertCommand	A SQL statement or stored procedure to insert new rows into an Oracle database
MissingMappingAction	Inherited from System.Data.Common.DataAdapter
MissingSchemaAction	Inherited from System.Data.Common.DataAdapter
Requery	Determines whether or not the SelectCommand is reexecuted on the next call to Fill
ReturnProviderSpecificTypes	Determines if the Fill method returns ODP.NET-specific values or .NET common language specification values
SafeMapping	Creates a mapping between column names in the result set to .NET types, to preserve the data <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
SelectCommand	A SQL statement or stored procedure that returns a single or multiple result set
Site	Inherited from System.ComponentModel.Component
SuppressGetDecimalInvalidCastException	Specifies whether to suppress the InvalidCastException and return a rounded-off 28 or 29 precision Oracle NUMBER value that can be represented as a .NET decimal.
TableMappings	Inherited from System.Data.Common.DataAdapter
UpdateBatchSize	Specifies a value that enables or disables batch processing support, and specifies the number of SQL statements that can be executed in a single round-trip to the database
UpdateCommand	A SQL statement or stored procedure to update rows from the DataSet to an Oracle database

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)

DeleteCommand

This property is a SQL statement or stored procedure to delete rows from an Oracle database.

Declaration

```
// C#  
public OracleCommand DeleteCommand {get; set;}
```

Property Value

An `OracleCommand` used during the `Update` call to delete rows from tables in the Oracle database, corresponding to the deleted rows in the `DataSet`.

Remarks

Default = `null`

If there is primary key information in the `DataSet`, the `DeleteCommand` can be automatically generated using the `OracleCommandBuilder`, if no command is provided for this.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)

IdentityInsert

When inserting `DataSet` data into the database, this property indicates whether the database generates the inserted row's identity column value or `DataSet` supplies this value.

Declaration

```
// C#  
public bool IdentityInsert {get; set;}
```

Property Value

When set to `true`, ODP.NET inserts `DataSet` identity column values into the database. When set to `false`, the database determines the inserted identity column values.

Remarks

This property applies only to identity columns of type GENERATED BY DEFAULT and GENERATED BY DEFAULT ON NULL. Identity column of type GENERATED ALWAYS will ignore this property and will always use database generated values.

When set to false, the server will generate an identity value for the row. That generated identity value returns back to the client to update the DataSet value.

When this property is set to true for the GENERATED BY DEFAULT case and the application attempts to insert a NULL value into the database's identity column, the NOT NULL constraint is violated and an error occurs. ODP.NET will then allow the database to generate the identity column value and return the generated value to the DataSet.

The default value for this property is false.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)
- [OracleIdentityType Enumeration](#)

IdentityUpdate

When updating DataSet data into the database, this property indicates whether to replace the database's identity column values with values of the DataSet or leave the current values unchanged.

Declaration

```
// C#  
public bool IdentityUpdate {get; set;}
```

Property Value

When set to true, ODP.NET updates the database identity column values with the values of the DataSet. When set to false, the database identity columns are left unchanged.

Remarks

This property applies only to identity columns of type GENERATED BY DEFAULT and GENERATED BY DEFAULT ON NULL. In the case of type GENERATED ALWAYS, this property will be ignored and the database will always retain its current identity values.

When set to false, the existing identity column value in the server is returned to the DataSet.

When this property is set to true for the GENERATED BY DEFAULT and GENERATED BY DEFAULT ON NULL cases and the application attempts to update the database's identity column with a NULL value, the NOT NULL constraint is violated and an error occurs. ODP.NET then does not update the identity column value and instead returns the existing identity column value of the database to the DataSet.

The default value for this property is false.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)
- [OracleIdentityType Enumeration](#)

InsertCommand

This property is a SQL statement or stored procedure to insert new rows into an Oracle database.

Declaration

```
// C#  
public OracleCommand InsertCommand {get; set;}
```

Property Value

An `OracleCommand` used during the `Update` call to insert rows into a table, corresponding to the inserted rows in the `DataSet`.

Remarks

Default = null

If there is primary key information in the `DataSet`, the `InsertCommand` can be automatically generated using the `OracleCommandBuilder`, if no command is provided for this property.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)

Requery

This property determines whether or not the `SelectCommand` is reexecuted on the next call to `Fill`.

Declaration

```
// C#  
public Boolean Requery {get; set;}
```

Property Value

Returns `true` if the `SelectCommand` is reexecuted on the next call to `Fill`; otherwise, returns `false`.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)
- ["OracleDataAdapter Requery Property"](#)

ReturnProviderSpecificTypes

This property determines if the Fill method returns ODP.NET-specific values or .NET common language specification compliant values.

Declaration

```
// C#  
public Boolean ReturnProviderSpecificTypes {get; set;}
```

Property Value

A value that indicates whether or not the Fill method returns ODP.NET-specific values.

Starting with ODP.NET 12.1.0.2, when set to true and LegacyEntireLOBFetch = 0 (default), BLOB and CLOB column values are represented in the DataTable as OracleBlob and OracleClob, respectively.

A value of false indicates that the Fill method returns .NET common language specification compliant values. The default is false.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)

SafeMapping

This property creates a mapping between column names in the result set to .NET types that represent column values in the DataSet, to preserve the data.

Declaration

```
// C#  
public Hashtable SafeMapping {get; set;}
```

Property Value

A hash table.

Remarks

Default = null

The SafeMapping property is used, when necessary, to preserve data in the following types:

- DATE
- TimeStamp (refers to all TimeStamp objects)
- INTERVAL DAY TO SECOND
- NUMBER

Example

See the example in "[OracleDataAdapter Safe Type Mapping](#)".

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)
- "[OracleDataAdapter Safe Type Mapping](#)"

SelectCommand

This property is a SQL statement or stored procedure that returns single or multiple result sets.

Declaration

```
// C#  
public OracleCommand SelectCommand {get; set;}
```

Property Value

An OracleCommand used during the Fill call to populate the selected rows to the DataSet.

Remarks

Default = null

If the SelectCommand does not return any rows, no tables are added to the dataset and no exception is raised.

If the SELECT statement selects from a VIEW, no key information is retrieved when a FillSchema() or a Fill() with MissingSchemaAction.AddWithKey is invoked.

Setting the SelectCommand property will also set the SuppressGetDecimalInvalidCastException property to the OracleCommand.Connection SuppressGetDecimalInvalidCastException property value, unless SuppressGetDecimalInvalidCastException property has been set explicitly.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)
- ["OracleDataAdapter Requery Property"](#)

SuppressGetDecimalInvalidCastException

This property specifies whether to suppress the `InvalidCastException` and return a rounded-off 28 or 29 precision Oracle NUMBER value that can be represented as a .NET decimal.

Declaration

```
// C#  
public bool SuppressGetDecimalInvalidCastException { get; set; }
```

Property Type

System.Boolean

Remarks

Oracle NUMBER has a maximum of 38 precision. .NET Decimal has a maximum of 28 or 29 precision. When the `GetDecimal()` method is called for an Oracle NUMBER value that cannot be represented as a .NET Decimal, then ODP.NET throws an `InvalidCastException` because not all the precision can be preserved when converting the number to a .NET Decimal.

This behavior occurs when `SuppressGetDecimalInvalidCastException` is set to false. Its default value is false.

When `SuppressGetDecimalInvalidCastException` is set to true, then the resulting decimal will be rounded off to 28 or 29 precision so that it can fit as a .NET decimal without throwing an exception. If the resulting rounded number is larger than can be stored in a .NET Decimal, an exception will be thrown, such as the number 1×10^{32} .

The `SuppressGetDecimalInvalidCastException` property can also be configured on `OracleConfiguration`, `OracleConnection`, and `OracleDataReader` objects. The `OracleDataAdapter SuppressGetDecimalInvalidCastException` property value inherits `OracleConnection`, `OracleCommand.Connection`, or `OracleConfiguration SuppressGetDecimalInvalidCastException` property value at construction time.

By using `OracleDataAdapter.SuppressGetDecimalInvalidCastException`, this ODP.NET setting becomes enabled for the `OracleDataAdapter`'s lifetime.

① See Also

- OracleConfiguration [SuppressGetDecimalInvalidCastException](#)
- OracleConnection [SuppressGetDecimalInvalidCastException](#)
- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)

UpdateBatchSize

This property specifies a value that enables or disables batch processing support, and specifies the number of SQL statements that can be executed in a single round-trip to the database.

Declaration

```
// C#  
public virtual int UpdateBatchSize {get; set;}
```

Property Value

An integer that returns the batch size.

Exceptions

`ArgumentOutOfRangeException` - The value is set to a number < 0 .

Remarks

Update batches executed with large amounts of data may encounter an "PLS-00123: Program too large" error. To avoid this error, reduce the size of `UpdateBatchSize` to a smaller value.

For each row in the `DataSet` that has been modified, added, or deleted, one SQL statement will be executed on the database.

Values are as follows:

- Value = 0
The data adapter executes all the SQL statements in a single database round-trip
- Value = 1 - Default value
This value disables batch updating and SQL statements are executed one at a time.
- Value = n where $n > 1$
The data adapter updates n rows of data per database round-trip.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)
- ["Batch Processing"](#)

UpdateCommand

This property is a SQL statement or stored procedure to update rows from the DataSet to an Oracle database.

Declaration

```
// C#  
public OracleCommand UpdateCommand {get; set;}
```

Property Value

An OracleCommand used during the Update call to update rows in the Oracle database, corresponding to the updated rows in the DataSet.

Remarks

Default = null

If there is primary key information in the DataSet, the UpdateCommand can be automatically generated using the OracleCommandBuilder, if no command is provided for this property.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)
- ["OracleDataAdapter Requery Property"](#)

OracleDataAdapter Public Methods

OracleDataAdapter public methods are listed in [Table 6-74](#).

Table 6-74 OracleDataAdapter Public Methods

Public Method	Description
CreateObjRef	Inherited from System.MarshalByRefObject
Dispose	Inherited from System.ComponentModel.Component
Equals	Inherited from System.Object (Overloaded)

Table 6-74 (Cont.) OracleDataAdapter Public Methods

Public Method	Description
Fill	Adds or refreshes rows in the DataSet to match the data in the Oracle database (Overloaded)
FillSchema	Inherited from System.Data.Common.DbDataAdapter
GetFillParameters	Inherited from System.Data.Common.DbDataAdapter
GetHashCode	Inherited from System.Object
GetLifetimeService	Inherited from System.MarshalByRefObject
GetType	Inherited from System.Object
InitializeLifetimeService	Inherited from System.MarshalByRefObject
ToString	Inherited from System.Object
Update	Inherited from System.Data.Common.DbDataAdapter

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)

Fill

Fill populates or refreshes the specified DataTable or DataSet.

Overload List:

- [Fill\(DataTable, OracleRefCursor\)](#)
This method adds or refreshes rows in the specified DataTable to match those in the provided OracleRefCursor object.
- [Fill\(DataSet, OracleRefCursor\)](#)
This method adds or refreshes rows in the DataSet to match those in the provided OracleRefCursor object.
- [Fill\(DataSet, string, OracleRefCursor\)](#)
This method adds or refreshes rows in the specified source table of the DataSet to match those in the provided OracleRefCursor object.
- [Fill\(DataSet, int, int, string, OracleRefCursor\)](#)
This method adds or refreshes rows in a specified range in the DataSet to match rows in the provided OracleRefCursor object.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)

Fill(DataTable, OracleRefCursor)

This method adds or refreshes rows in the specified DataTable to match those in the provided OracleRefCursor object.

Declaration

```
// C#  
public int Fill(DataTable dataTable, OracleRefCursor refCursor);
```

Parameters

- *dataTable*
The DataTable object being populated.
- *refCursor*
The OracleRefCursor that rows are being retrieved from.

Return Value

The number of rows added to or refreshed in the DataTable.

Exceptions

ArgumentNullException - The *dataTable* or *refCursor* parameter is null.

InvalidOperationException - The OracleRefCursor is already being used to fetch data.

NotSupportedException - The SafeMapping type is not supported.

Remarks

No schema or key information is provided, even if the Fill method is called with MissingSchemaAction set to MissingSchemaAction.AddWithKey.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)
- ["OracleDataAdapter Requery Property"](#)

Fill(DataSet, OracleRefCursor)

This method adds or refreshes rows in the DataSet to match those in the provided OracleRefCursor object.

Declaration

```
// C#  
public int Fill(DataSet dataSet, OracleRefCursor refCursor);
```

Parameters

- *dataSet*
The DataSet object being populated.
- *refCursor*
The OracleRefCursor that rows are being retrieved from.

Return Value

Returns the number of rows added or refreshed in the DataSet.

Exceptions

ArgumentNullException - The *dataSet* or *refCursor* parameter is null.

InvalidOperationException - The OracleRefCursor is already being used to fetch data.

InvalidOperationException - The OracleRefCursor is ready to fetch data.

NotSupportedException - The SafeMapping type is not supported.

Remarks

If there is no DataTable to refresh, a new DataTable named Table is created and populated using the provided OracleRefCursor object.

No schema or key information is provided, even if the Fill method is called with MissingSchemaAction set to MissingSchemaAction.AddWithKey.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)
- ["OracleDataAdapter Requery Property"](#)

Fill(DataSet, string, OracleRefCursor)

This method adds or refreshes rows in the specified source table of the DataSet to match those in the provided OracleRefCursor object.

Declaration

```
// C#  
public int Fill(DataSet dataSet, string srcTable, OracleRefCursor  
    refCursor);
```

Parameters

- *dataSet*
The DataSet object being populated.
- *srcTable*
The name of the source table used in the table mapping.
- *refCursor*
The OracleRefCursor that rows are being retrieved from.

Return Value

Returns the number of rows added or refreshed into the DataSet.

Exceptions

ArgumentNullException - The *dataSet* or *refCursor* parameter is null.

InvalidOperationException - The OracleRefCursor is already being used to fetch data or the source table name is invalid.

NotSupportedException - The SafeMapping type is not supported.

Remarks

No schema or key information is provided, even if the Fill method is called with MissingSchemaAction set to MissingSchemaAction.AddWithKey.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)
- ["OracleDataAdapter Requery Property"](#)

Fill(DataSet, int, int, string, OracleRefCursor)

This method adds or refreshes rows in a specified range in the DataSet to match rows in the provided OracleRefCursor object.

Declaration

```
// C#  
public int Fill(DataSet dataSet, int startRecord, int maxRecords,  
    string srcTable, OracleRefCursor refCursor);
```

Parameters

- *dataSet*
The DataSet object being populated.
- *startRecord*
The record number to start with.
- *maxRecords*
The maximum number of records to obtain.
- *srcTable*
The name of the source table used in the table mapping.
- *refCursor*
The OracleRefCursor that rows are being retrieved from.

Return Value

This method returns the number of rows added or refreshed in the DataSet. This does not include rows affected by statements that do not return rows.

Exceptions

ArgumentNullException - The *dataSet* or *refCursor* parameter is null.

InvalidOperationException - The OracleRefCursor is already being used to fetch data or the source table name is invalid.

NotSupportedException - The SafeMapping type is not supported.

Remarks

No schema or key information is provided, even if the Fill method is called with MissingSchemaAction set to MissingSchemaAction.AddWithKey.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)
- "[OracleDataAdapter Requery Property](#)"

OracleDataAdapter Events

OracleDataAdapter events are listed in [Table 6-75](#).

Table 6-75 OracleDataAdapter Events

Event Name	Description
Disposed	Inherited from System.ComponentModel.Component
FillError	Inherited from System.Data.Common.DbDataAdapter
RowUpdated	This event is raised when row(s) have been updated by the Update () method
RowUpdating	This event is raised when row data are about to be updated to the database

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)

RowUpdated

This event is raised when row(s) have been updated by the Update () method.

Declaration

```
// C#
public event OracleRowUpdatedEventHandler RowUpdated;
```

Event Data

The event handler receives an OracleRowUpdatedEventArgs object which exposes the following properties containing information about the event.

- **Command**
The OracleCommand executed during the Update.
- **Errors** (inherited from RowUpdatedEventArgs)
The exception, if any, is generated during the Update.
- **RecordsAffected** (inherited from RowUpdatedEventArgs)
The number of rows modified, inserted, or deleted by the execution of the Command.
- **Row** (inherited from RowUpdatedEventArgs)
The DataRow sent for Update.
- **StatementType** (inherited from RowUpdatedEventArgs)
The type of SQL statement executed.
- **Status** (inherited from RowUpdatedEventArgs)
The UpdateStatus of the Command.
- **TableMapping** (inherited from RowUpdatedEventArgs)
The DataTableMapping used during the Update.

Example

The following example shows how to use the RowUpdating and RowUpdated events.

```
// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;

class RowUpdatedSample
{
    // Event handler for RowUpdating event
    protected static void OnRowUpdating(object sender,
                                        OracleRowUpdatingEventArgs e)
    {
        Console.WriteLine("Row updating....");
        Console.WriteLine("Event arguments:");
        Console.WriteLine("Command Text: " + e.Command.CommandText);
        Console.WriteLine("Command Type: " + e.StatementType);
        Console.WriteLine("Status: " + e.Status);
    }

    // Event handler for RowUpdated event
    protected static void OnRowUpdated(object sender,
                                       OracleRowUpdatedEventArgs e)
    {
        Console.WriteLine("Row updated....");
        Console.WriteLine("Event arguments:");
        Console.WriteLine("Command Text: " + e.Command.CommandText);
        Console.WriteLine("Command Type: " + e.StatementType);
        Console.WriteLine("Status: " + e.Status);
    }

    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        string cmdstr = "SELECT EMPNO, ENAME, SAL FROM EMP";

        // Create the adapter with the selectCommand txt and the
        // connection string
        OracleDataAdapter adapter = new OracleDataAdapter(cmdstr, constr);

        // Create the builder for the adapter to automatically generate
        // the Command when needed
        OracleCommandBuilder builder = new OracleCommandBuilder(adapter);

        // Create and fill the DataSet using the EMP
        DataSet dataset = new DataSet();
        adapter.Fill(dataset, "EMP");

        // Get the EMP table from the dataset
        DataTable table = dataset.Tables["EMP"];

        // Indicate DataColumn EMPNO is unique
        // This is required by the OracleCommandBuilder to update the EMP table
        table.Columns["EMPNO"].Unique = true;

        // Get the first row from the EMP table
        DataRow row = table.Rows[0];

        // Update the salary
```

```
double sal = double.Parse(row["SAL"].ToString());
row["SAL"] = sal + .01;

// Set the event handlers for the RowUpdated and the RowUpdating event
// the OnRowUpdating() method will be triggered before the update, and
// the OnRowUpdated() method will be triggered after the update
adapter.RowUpdating += new OracleRowUpdatingEventHandler(OnRowUpdating);
adapter.RowUpdated += new OracleRowUpdatedEventHandler(OnRowUpdated);

// Now update the EMP using the adapter
// The OracleCommandBuilder will create the UpdateCommand for the
// adapter to update the EMP table
// The OnRowUpdating() and the OnRowUpdated() methods will be triggered
adapter.Update(dataset, "EMP");
}
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)
- ["OracleRowUpdatedEventHandler Delegate"](#)

RowUpdating

This event is raised when row data are about to be updated to the database.

Declaration

```
// C#
public event OracleRowUpdatingEventHandler RowUpdating;
```

Event Data

The event handler receives an `OracleRowUpdatingEventArgs` object which exposes the following properties containing information about the event.

- **Command**
The `OracleCommand` executed during the Update.
- **Errors** (inherited from `RowUpdatingEventArgs`)
The exception, if any, is generated during the Update.
- **Row** (inherited from `RowUpdatingEventArgs`)
The `DataRow` sent for Update.
- **StatementType** (inherited from `RowUpdatingEventArgs`)
The type of SQL statement executed.
- **Status** (inherited from `RowUpdatingEventArgs`)
The `UpdateStatus` of the Command.
- **TableMapping** (inherited from `RowUpdatingEventArgs`)

The DataTableMapping used during the Update.

Example

The example for the RowUpdated event also shows how to use the RowUpdating event. See RowUpdated event "[Example](#)".

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDataAdapter Class](#)
- [OracleDataAdapter Members](#)
- "[OracleRowUpdatingEventHandler Delegate](#)"

OracleDatabase Class

An OracleDatabase object represents an Oracle Database instance.

Class Inheritance

System.Object

Oracle.DataAccess.Client.OracleDatabase

Declaration

```
// C#
public sealed class OracleDatabase : IDisposable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#

using System;
```

```

using Oracle.DataAccess.Client;

namespace Startup
{
    class Test
    {
        static void Main()
        {
            OracleConnection con = null;
            OracleDatabase db = null;
            string constring = "dba privilege=sysdba;user id=scott;password=tiger;data
source=oracle";

            try
            {
                // Open a connection to see if the DB is up
                con = new OracleConnection(constring);
                con.Open();

                Console.WriteLine("The Oracle database is already up.");
            }
            catch (OracleException ex)
            {
                // If the database is down, start up the DB
                if (ex.Number == 1034)
                {
                    Console.WriteLine("The Oracle database is down.");

                    // Create an instance of an OracleDatabase object
                    db = new OracleDatabase(constring);

                    // Start up the database
                    db.Startup();

                    Console.WriteLine("The Oracle database is now up.");

                    // Executing Startup() is the same as the following:
                    // db.Startup(OracleDBStartupMode.NoRestriction, null, true);
                    // which is also the same as:
                    // db.Startup(OracleDBStartupMode.NoRestriction, null, false);
                    // db.ExecuteNonQuery("ALTER DATABASE MOUNT");
                    // db.ExecuteNonQuery("ALTER DATABASE OPEN");

                    // Dispose the OracleDatabase object
                    db.Dispose();
                }
                else
                {
                    Console.WriteLine("Error: " + ex.Message);
                }
            }
            finally
            {
                // Dispose the OracleConnetion object
                con.Dispose();
            }
        }
    }
}

```


① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDatabase Members](#)
- [OracleDatabase Constructors](#)
- [OracleDatabase Properties](#)
- [OracleDatabase Public Methods](#)

OracleDatabase Members

OracleDatabase members are listed in the following tables.

OracleDatabase Constructors

The OracleDatabase constructor is listed in [Table 6-76](#).

Table 6-76 OracleDatabase Constructors

Constructor	Description
OracleDatabase Constructors	Instantiates a new instance of OracleDatabase class

OracleDatabase Properties

The OracleDatabase properties are listed in [Table 6-77](#).

Table 6-77 OracleDatabase Properties

Property	Description
ServerVersion	Specifies the database version number of the Oracle Database instance to which the connection is made

OracleDatabase Public Methods

The OracleDatabase public methods are listed in [Table 6-78](#).

Table 6-78 OracleDatabase Public Methods

Public Method	Description
Dispose	Releases any resources or memory allocated by the object.
ExecuteNonQuery	Executes the supplied non-SELECT statement against the database
Shutdown	Shuts down the database (Overloaded)
Startup	Starts up the database (Overloaded)

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDatabase Class](#)

OracleDatabase Constructors

The OracleDatabase constructors instantiate a new instance of the OracleDatabase class.

Overload List

- [OracleDatabase Constructor\(string\)](#)
The OracleDatabase constructor instantiates a new instance of the OracleDatabase class using the supplied connection string.
- [OracleDatabase Constructor\(string, OracleCredential\)](#)
The OracleDatabase constructor instantiates a new instance of the OracleDatabase class using the supplied connection string and OracleCredential object .

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDatabase Class](#)
- [OracleDatabase Members](#)

OracleDatabase Constructor(string)

The OracleDatabase constructor instantiates a new instance of the OracleDatabase class using the supplied connection string.

Declaration

```
// C#  
public OracleDatabase(string connectionString);
```

Parameters

- *connectionString*

The connection information used to connect to the Oracle Database instance.

Remarks

The *connectionString* follows the same format used by the OracleConnection object. However, the OracleDatabase constructor accepts only the user_id, password, data source, and dba_privilege connection string attributes. All other attribute values are ignored. The supplied *connectionString* must contain the dba_privilege connection string attribute that is set to SYSDBA, SYSASM, SYSOPER, or another valid administrator privilege.

The `OracleDatabase` object creates a connection upon construction and remains connected throughout its lifetime. The connection is destroyed when the `OracleDatabase` object is disposed. This connection is not pooled to be used by another `OracleDatabase` object.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDatabase Class](#)
- [OracleDatabase Members](#)

OracleDatabase Constructor(string, OracleCredential)

The `OracleDatabase` constructor instantiates a new instance of the `OracleDatabase` class using the supplied connection string and `OracleCredential` object.

Declaration

```
// C#  
public OracleDatabase(string connectionString, OracleCredential orclCredential)
```

Parameters

- *connectionString*
connection string that does not contain any of user id, password, DBA Privilege, proxy user id, nor proxy password.
- *orclCredential*
`OracleCredential` object containing user credentials. If this parameter is passed as null then `OracleConnection`'s behavior will be same as `OracleConnection` with normal connection string.

Exceptions

`InvalidOperationException` is raised when non-null `OracleCredential` object is used with a connection string containing any of user id, password, DBA Privilege, proxy user id, or proxy password.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDatabase Class](#)
- [OracleDatabase Members](#)

OracleDatabase Properties

The `OracleDatabase` properties are listed in [Table 6-79](#).

Table 6-79 OracleDatabase Properties

Property	Description
ServerVersion	Specifies the database version number of the Oracle Database instance to which the connection is made

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDatabase Class](#)
- [OracleDatabase Members](#)

ServerVersion

This property returns the database version number of the Oracle Database instance to which the connection is made.

Declaration

```
Public string ServerVersion {get;}
```

Property value

Returns the database version of the Oracle Database instance.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDatabase Class](#)
- [OracleDatabase Members](#)

OracleDatabase Public Methods

The OracleDatabase public methods are listed in [Table 6-80](#).

Table 6-80 OracleDatabase Public Methods

Public Method	Description
Dispose	Releases any resources or memory allocated by the object.
ExecuteNonQuery	Executes the supplied non-SELECT statement against the database
Shutdown	Shuts down the database (Overloaded)
Startup	Starts up the database (Overloaded)

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDatabase Class](#)
- [OracleDatabase Members](#)

Dispose

This method releases any resources or memory allocated by the object.

Declaration

```
// C#  
public void Dispose();
```

Implements

IDisposable

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDatabase Class](#)
- [OracleDatabase Members](#)

ExecuteNonQuery

This method executes the supplied non-SELECT statement against the database.

Declaration

```
// C#  
public void ExecuteNonQuery(string sql);
```

Exceptions

OracleException - The command execution has failed.

Remarks

This method is meant for execution of DDL statements such as ALTER DATABASE statements to OPEN and MOUNT the database, for example. This method should not be used to execute SQL SELECT statements. This method does not support any parameter binding.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDatabase Class](#)
- [OracleDatabase Members](#)

Shutdown

Shutdown methods shut down a database instance.

Overload List

- [Shutdown\(\)](#)
This method shuts down the database.
- [Shutdown\(OracleDBShutdownMode, bool\)](#)
This method shuts down the database using the specified mode.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDatabase Class](#)
- [OracleDatabase Members](#)
- ["OracleDBShutdownMode Enumeration"](#)

Shutdown()

This method shuts down the database.

Declaration

```
// C#  
public void Shutdown();
```

Exceptions

OracleException - The database shutdown request has failed.

Remarks

This method shuts down a database instance in the `OracleDBShutdownMode.Default` mode. New connections are refused, and the method waits for the existing connections to end.

Note

As the shutdown is effected using the `OracleDBShutdownMode.Default` mode, the shutdown request may remain pending if there are open connections other than the connection created by the `OracleDatabase` object.

After the connections have closed, the method closes the database, dismounts the database, and shuts down the instance using the `OracleDBShutdownMode.Final` mode.

This method does not throw exceptions for cases where the database has been already closed, dismounted, or shutdown appropriately. If other errors are encountered, then an exception is thrown.

Invoking this method against an Oracle Real Application Clusters (Oracle RAC) database shuts down only that database instance to which the `OracleDatabase` object is connected.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDatabase Class](#)
- [OracleDatabase Members](#)
- ["OracleDBShutdownMode Enumeration"](#)

Shutdown(OracleDBShutdownMode, bool)

This method shuts down the database instance using the specified mode.

Declaration

```
//C#  
public void Shutdown(OracleDBShutdownMode shutdownMode, bool bCloseDismountAndFinalize);
```

Parameters

- *shutdownMode*
A `OracleDBShutdownMode` enumeration value.
- *bCloseDismountAndFinalize*
A boolean signifying whether the database is to be closed, dismounted, and finalized.

Exceptions

`OracleException` - The database shutdown request has failed.

Remarks

This method shuts down a database instance in the specified mode. If the *bCloseDismountAndFinalize* parameter is true, then the method also closes the database, dismounts the database, and shuts down the instance using the `OracleDBShutdownMode.Final` mode.

If the *bCloseDismountAndFinalize* parameter is true, then this method does not throw exceptions for cases where the database has been already closed, dismounted, or shutdown appropriately. If other errors are encountered, then an exception is thrown.

If the *bCloseDismountAndFinalize* parameter is false, then the application needs to explicitly close and dismount the database. The application can then reinvoke the method using the `OracleDBShutdownMode.Final` mode to properly shut down the database. For example, if `db` is an instance of the `OracleDatabase` class, then the application invokes the following:

1. `db.Shutdown(OracleDBShutdownMode.Default, false);`
2. `db.ExecuteNonQuery("ALTER DATABASE CLOSE NORMAL");`
3. `db.ExecuteNonQuery("ALTER DATABASE DISMOUNT");`
4. `db.Shutdown(OracleDBShutdownMode.Final);`

Note

- The `OracleDBShutdownMode.Final` enumeration value should not be used as the *shutdownMode* for the initial method invocation. The `OracleDBShutdownMode.Final` mode should be used only if the database is already closed and dismounted. Otherwise, the method might wait indefinitely.
- If the specified *shutdownMode* is `OracleDBShutdownMode.Final`, then the value of the *bCloseDismountAndFinalize* input parameter is ignored, as the database should have been closed and dismounted already.

If the specified *shutdownMode* is `OracleDBShutdownMode.Abort`, then the value of the *bCloseDismountAndFinalize* input parameter is ignored, as the `Abort` mode requires the database to be closed, dismounted, and finalized.

Invoking this method against an Oracle Real Application Clusters (Oracle RAC) database shuts down only that database instance to which the `OracleDatabase` object is connected.

Example

```
using System;
using Oracle.DataAccess.Client;

namespace Shutdown
{
    class Test
    {
        static void Main()
        {
            OracleConnection con = null;
            OracleDatabase db = null;
            string constring = "user id=scott;password=tiger;data source=oracle;" +
                "pooling=false;dba privilege=sysdba";

            try
            {
                // Open a connection to see if the DB is up;
                con = new OracleConnection(constring);
                con.Open();

                Console.WriteLine("The Oracle database is currently up.");
            }
        }
    }
}
```



```

        // If open succeeds, we know that the database is up.
        // We have to dispose the connection so that we can
        // shutdown the database.
        con.Dispose();

        // Shutdown the database
        db = new OracleDatabase(constring);
        db.Shutdown();

        Console.WriteLine("The Oracle database is shut down.");

        // Executing Shutdown() above is the same as the following:
        // db.Shutdown(OracleDBShutdownMode.Default, false);
        // db.ExecuteNonQuery("ALTER DATABASE CLOSE NORMAL");
        // db.ExecuteNonQuery("ALTER DATABASE DISMOUNT");
        // db.Shutdown(OracleDBShutdownMode.Final);

        // Dispose the OracleDatabase object
        db.Dispose();
    }
    catch (OracleException ex)
    {
        Console.WriteLine("An error has occurred: {0}", ex.Message);
    }
}
}
}

```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDatabase Class](#)
- [OracleDatabase Members](#)
- ["OracleDBShutdownMode Enumeration"](#)

Startup

Startup methods enable a user with database administrator privileges to start a database instance.

Overload List

- [Startup\(\)](#)
This method starts a database instance using the server-side parameter file.
- [Startup\(OracleDBStartupMode, bool\)](#)
This method starts up the database using the specified startup mode.
Not Available in ODP.NET, Unmanaged Driver
- [Startup\(OracleDBStartupMode, string, bool\)](#)
This method starts a database instance using the client-side parameter file.
Not Available in ODP.NET, Managed Driver and ODP.NET Core

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDatabase Class](#)
- [OracleDatabase Members](#)
- ["OracleDBStartupMode Enumeration"](#)

Startup()

This method starts up the database.

Declaration

```
// C#  
public void Startup();
```

Exceptions

OracleException - The database startup request has failed.

Remarks

This method starts a database instance in the `OracleDbStartupMode.Normal` mode using the server-side parameter file (spfile). After the database is successfully started, this method also executes the `ALTER DATABASE MOUNT` and `ALTER DATABASE OPEN` statements.

This method does not throw exceptions for cases where the database is already mounted, opened, or started appropriately. If other errors are encountered, then an exception is thrown.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDatabase Class](#)
- [OracleDatabase Members](#)
- ["OracleDBStartupMode Enumeration"](#)

Startup(OracleDBStartupMode, bool)

This method starts up the database using the specified startup mode.

Declaration

```
// C#  
public void Startup(OracleDbStartupMode startupMode, bool bMountAndOpen);
```

Parameters

startupMode - An `OracleDBStartupMode` enumeration value.

`bMountAndOpen` - A true/false value signifying whether the database is to be mounted and opened.

Exceptions

`OracleException` - The database startup request has failed.

Remarks

This method starts a database instance in the specified mode. After the database is successfully started, and if `bMountAndOpen` input parameter is true, this method also executes the `ALTER DATABASE MOUNT` and `ALTER DATABASE OPEN` statements.

If `bMountAndOpen` is true, then this method does not throw an exception for cases where the database is already mounted, opened, or started appropriately. If other errors are encountered, then an exception is thrown.

If `bMountAndOpen` is false, then the database must be mounted and opened explicitly by the application. For example, if `db` is an instance of the `OracleDatabase` class, then the application invokes the following:

- `db.Startup(OracleDBStartupMode.NoRestriction, false);`
- `db.ExecuteNonQuery("ALTER DATABASE MOUNT");`
- `db.ExecuteNonQuery("ALTER DATABASE OPEN");`

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDatabase Class](#)
- [OracleDatabase Members](#)

Startup(OracleDBStartupMode, string, bool)

This method starts up the database using the specified startup mode.

Declaration

```
// C#  
public void Startup(OracleDbStartupMode startupMode, string pfile, bool bMountAndOpen);
```

Parameters

- *startupMode*
An `OracleDBStartupMode` enumeration value.
- *pfile*
The location and name of the client-side parameter file. For example, "c:\\admin\\init.ora".

The name of the parameter file varies depending on the operating system. For example, it can be in mixed case or lowercase, or it can have a logical name or a variation of the name `init.ora`. The default location is usually `ORACLE_HOME/dbs` or `ORACLE_HOME\\database`.
- *bMountAndOpen*

A true/false value signifying whether the database is to be mounted and opened.

Exceptions

OracleException - The database startup request has failed.

Remarks

This method starts a database instance in the specified mode using the specified client-side parameter file. After the database is successfully started, and if *bMountAndOpen* input parameter is true, this method also executes the ALTER DATABASE MOUNT and ALTER DATABASE OPEN statements.

If *bMountAndOpen* is true, then this method does not throw an exception for cases where the database is already mounted, opened, or started appropriately. If other errors are encountered, then an exception is thrown.

If *bMountAndOpen* is false, then the database must be mounted and opened explicitly by the application. For example, if *db* is an instance of the *OracleDatabase* class, then the application invokes the following:

1. `db.Startup(OracleDBStartupMode.NoRestriction, null, false);`
2. `db.ExecuteNonQuery("ALTER DATABASE MOUNT");`
3. `db.ExecuteNonQuery("ALTER DATABASE OPEN");`

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDatabase Class](#)
- [OracleDatabase Members](#)
- ["OracleDBStartupMode Enumeration"](#)

OracleDataReader Class

An *OracleDataReader* object represents a forward-only, read-only, in-memory result set.

Unlike the *DataSet*, the *OracleDataReader* object stays connected and fetches one row at a time.

The following section contain related information:

- ["Obtaining LONG and LONG RAW Data"](#).
- ["Obtaining Data from an OracleDataReader Object"](#).

Class Inheritance

System.Object

System.MarshalByRefObject

System.Data.Common.DataReader

System.Data.Common.DbDataReader

Oracle.DataAccess.Client.OracleDataReader

Declaration

```
// C#
public sealed class OracleDataReader : DbDataReader, IEnumerable,
    IDataReader, IDisposable, IDataRecord
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

An OracleDataReader instance is constructed by a call to the ExecuteReader method of the OracleCommand object. The only properties that can be accessed after the DataReader is closed or has been disposed, are IsClosed and RecordsAffected.

To minimize the number of open database cursors, OracleDataReader objects should be explicitly disposed.

Example

The following OracleDataReader example retrieves the data from the EMP table:

```
/* Database Setup, if you have not done so yet.
connect scott/tiger@oracle
CREATE TABLE empInfo (
empno NUMBER(4) PRIMARY KEY,
empName VARCHAR2(20) NOT NULL,
hiredate DATE,
salary NUMBER(7,2),
jobDescription Clob,
byteCodes BLOB
);

Insert into empInfo(EMPNO,EMPNAME,JOBDESCRIPTION,byteCodes) values
(1,'KING','SOFTWARE ENGR', '5657');
Insert into empInfo(EMPNO,EMPNAME,JOBDESCRIPTION,byteCodes) values
(2,'SCOTT','MANAGER', '5960');
commit;

*/

// C#
```

```
using System;
using System.Data;
using Oracle.DataAccess.Client;

class OracleDataReaderSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        string cmdstr = "SELECT * FROM EMPINFO";
        OracleConnection connection = new OracleConnection(constr);
        OracleCommand cmd = new OracleCommand(cmdstr, con);

        OracleDataReader reader = cmd.ExecuteReader();

        // Declare the variables to retrieve the data in EmpInfo
        short empNo;
        string empName;
        DateTime hireDate;
        double salary;
        string jobDesc;
        byte[] byteCodes = new byte[10];

        // Read the next row until end of row
        while (reader.Read())
        {
            empNo = reader.GetInt16(0);
            Console.WriteLine("Employee number: " + empNo);
            empName = reader.GetString(1);
            Console.WriteLine("Employee name: " + empName);

            // The following columns can have NULL value, so it
            // is important to call IsDBNull before getting the column data
            if (!reader.IsDBNull(2))
            {
                hireDate = reader.GetDateTime(2);
                Console.WriteLine("Hire date: " + hireDate);
            }

            if (!reader.IsDBNull(3))
            {
                salary = reader.GetDouble(3);
                Console.WriteLine("Salary: " + salary);
            }

            if (!reader.IsDBNull(4))
            {
                jobDesc = reader.GetString(4);
                Console.WriteLine("Job Description: " + jobDesc);
            }

            if (!reader.IsDBNull(5))
            {
                long len = reader.GetBytes(5, 0, byteCodes, 0, 10);

                Console.Write("Byte codes: ");
                for (int i = 0; i < len; i++)
                    Console.Write(byteCodes[i].ToString("x"));
            }
        }
    }
}
```

```
        Console.WriteLine();  
    }  
  
    Console.WriteLine();  
}  
  
// Clean up  
reader.Dispose();  
con.Dispose();  
}  
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Members](#)
- [OracleDataReader Static Methods](#)
- [OracleDataReader Properties](#)
- [OracleDataReader Public Methods](#)

OracleDataReader Members

OracleDataReader members are listed in the following tables.

OracleDataReader Static Methods

The OracleDataReader static method is listed in [Table 6-81](#).

Table 6-81 OracleDataReader Static Method

Method	Description
Equals	Inherited from System.Object (Overloaded)

OracleDataReader Properties

OracleDataReader properties are listed in [Table 6-82](#).

Table 6-82 OracleDataReader Properties

Property	Description
Depth	Gets a value indicating the depth of nesting for the current row
FetchSize	Specifies the size of OracleDataReader's internal cache
FieldCount	Gets the number of columns in the result set
HasRows	Indicates whether the OracleDataReader has one or more rows
HiddenFieldCount	Gets the number of fields in the OracleDataReader that are hidden
IsClosed	Indicates whether or not the data reader is closed

Table 6-82 (Cont.) OracleDataReader Properties

Property	Description
Item	Gets the value of the column (Overloaded)
InitialLOBFetchSize	Specifies the amount that the OracleDataReader initially fetches for LOB columns
InitialLONGFetchSize	Specifies the amount that the OracleDataReader initially fetches for LONG and LONG RAW columns
RecordsAffected	Gets the number of rows changed, inserted, or deleted by execution of the SQL statement
RowSize	Gets the amount of memory the internal cache of the OracleDataReader needs to store one row of data.
RowsToFetchPerRoundTrip	Specifies the total number of rows to retrieve per database round trip <i>Not Available in ODP.NET, Unmanaged Driver</i>
SuppressGetDecimalInvalidCastException	Specifies whether to suppress the InvalidCastException and return a rounded-off 28 precision value if the Oracle NUMBER value has more than 28 precision.
UseEdmMapping	Indicates whether or not the OracleDataReader utilizes the Entity Data Model mapping configuration when returning values
VisibleFieldCount	Gets the number of fields in the OracleDataReader that are not hidden

OracleDataReader Public Methods

OracleDataReader public methods are listed in [Table 6-83](#).

Table 6-83 OracleDataReader Public Methods

Public Method	Description
Close	Closes the OracleDataReader
CreateObjRef	Inherited from System.MarshalByRefObject
Dispose	Releases any resources or memory allocated by the object
Equals	Inherited from System.Object (Overloaded)
GetBoolean	Returns the bool value of the specified numeric column
GetByte	Returns the byte value of the specified column
GetByteArray(int)	Returns the byte array value of the specified numeric column <i>Not Available in ODP.NET, Unmanaged Driver</i>
GetByteArray(string)	Returns the byte array value of the specified numeric column <i>Not Available in ODP.NET, Unmanaged Driver</i>
GetBytes	Populates the provided byte array with up to the maximum number of bytes, from the specified offset (in bytes) of the column
GetChar	<i>Not Supported</i>
GetChars	Populates the provided character array with up to the maximum number of characters, from the specified offset (in characters) of the column

Table 6-83 (Cont.) OracleDataReader Public Methods

Public Method	Description
GetData	<i>Not Supported</i>
GetDataTypeName	Returns the ODP.NET type name of the specified column
GetDateTime	Returns the DateTime value of the specified column
GetDateTimeOffset	Returns the DateTimeOffset value of the specified column
GetDecimal	Returns the decimal value of the specified NUMBER column
GetDouble	Returns the double value of the specified NUMBER column or BINARY_DOUBLE column
GetDoubleArray(int)	Returns the double[] value of the specified VECTOR column
GetDoubleArray(string)	Returns the double[] value of the specified VECTOR column
GetEnumerator	Returns an IEnumerator that can be used to iterate through the collection
GetFieldType	Returns the Type of the specified column
GetFieldValueAsync	Returns a Task-based asynchronous version of <code>OracleDataReader.GetFieldValue()</code> , which returns the column value as a specified type (Overloaded)
GetFloat	Returns the float value of the specified NUMBER column or BINARY_FLOAT column
GetFloatArray(int)	Returns the float[] value of the specified VECTOR column
GetFloatArray(string)	Returns the float[] value of the specified VECTOR column
GetGuid	Returns the Guid value of the specified binary data column
<code>GetHashCode</code>	Inherited from <code>System.Object</code>
GetInt16	Returns the Int16 value of the specified NUMBER column
GetInt16Array(int)	Returns the short[] value of the specified VECTOR column
GetInt16Array(string)	Returns the short[] value of the specified VECTOR column
GetInt32	Returns the Int32 value of the specified NUMBER column
GetInt64	Returns the Int64 value of the specified NUMBER column
<code>GetLifetimeService</code>	Inherited by <code>System.MarshalByRefObject</code>
GetName	Returns the name of the specified column
GetNonZeroDoubleArray	Returns the non-zero-dimension values double[] of the specified sparse VECTOR column.(Overloaded) <i>Not Available in ODP.NET, Unmanaged Driver</i>
GetNonZeroFloatArray	Returns the non-zero-dimension values float[] of the specified sparse VECTOR column.(Overloaded) <i>Not Available in ODP.NET, Unmanaged Driver</i>
GetNonZeroIndexArray	Returns the non-zero-dimension indices long[] of the specified sparse VECTOR column.(Overloaded) <i>Not Available in ODP.NET, Unmanaged Driver</i>
GetNonZeroInt16Array	Returns the non-zero-dimension values short[] of the specified sparse VECTOR column. (Overloaded) <i>Not Available in ODP.NET, Unmanaged Driver</i>

Table 6-83 (Cont.) OracleDataReader Public Methods

Public Method	Description
GetOracleBFile	Returns an <code>OracleBFile</code> object of the specified BFILE column
GetOracleBinary	Returns an <code>OracleBinary</code> structure of the specified column
GetOracleBlob	Returns an <code>OracleBlob</code> object of the specified BLOB column
GetOracleBlobForUpdate	Returns an updatable <code>OracleBlob</code> object of the specified BLOB column <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
GetOracleBoolean(int)	Returns the <code>OracleBoolean</code> value of the column
GetOracleClob	Returns an <code>OracleClob</code> object of the specified CLOB column
GetOracleClobForUpdate	Returns an updatable <code>OracleClob</code> object of the specified CLOB column <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
GetOracleDataReader	Returns the <code>OracleDataReader</code> object corresponding to the REF CURSOR type column in the select list <i>Not Available in ODP.NET, Unmanaged Driver</i>
GetOracleDate	Returns an <code>OracleDate</code> structure of the specified DATE column
GetOracleDecimal	Returns an <code>OracleDecimal</code> structure of the specified NUMBER column
GetOracleIntervalDS	Returns an <code>OracleIntervalDS</code> structure of the specified INTERVAL DAY TO SECOND column
GetOracleIntervalYM	Returns an <code>OracleIntervalYM</code> structure of the specified INTERVAL YEAR TO MONTH column
GetOracleRef	Returns an <code>OracleRef</code> object of the specified REF column
GetOracleRefCursor	Returns the <code>OracleRefCursor</code> object corresponding to the REF CURSOR type column in the select list <i>Not Available in ODP.NET, Unmanaged Driver</i>
GetOracleString	Returns an <code>OracleString</code> structure of the specified column
GetOracleTimeStamp	Returns an <code>OracleTimeStamp</code> structure of the Oracle TimeStamp column
GetOracleTimeStampLTZ	Returns an <code>OracleTimeStampLTZ</code> structure of the specified Oracle TimeStamp WITH LOCAL TIME ZONE column
GetOracleTimeStampTZ	Returns an <code>OracleTimeStampTZ</code> structure of the specified Oracle TimeStamp WITH TIME ZONE column
GetOracleXmlType	Returns an <code>OracleXmlType</code> object of the specified XMLType column
GetOracleValue	Returns the specified column value as a ODP.NET type
GetOracleValues	Gets all the column values as ODP.NET types
GetOracleVector	Returns <code>OracleVector</code> object of the specified VECTOR column. (Overloaded) <i>Not Available in ODP.NET, Unmanaged Driver</i>
GetOrdinal	Returns the 0-based ordinal (or index) of the specified column name
GetProviderSpecificFieldType	Returns the provider-specific type of the specified column

Table 6-83 (Cont.) OracleDataReader Public Methods

Public Method	Description
GetProviderSpecificValue	Returns an object that represents the underlying provider-specific value of the specified ordinal
GetProviderSpecificValues	Returns an array of objects that represent the underlying provider-specific values
GetSchemaTable	Returns a DataTable that describes the column metadata of the OracleDataReader
GetString	Returns the string value of the specified column
GetTimeSpan	Returns the TimeSpan value of the specified INTERVAL DAY TO SECOND column
GetType	Inherited from System.Object class
GetValue	Returns the column value as a .NET type
GetValues	Gets all the column values as .NET types
GetVectorDimensions	Returns the number of dimensions of the specified VECTOR column.(Overloaded) <i>Not Available in ODP.NET, Unmanaged Driver</i>
GetVectorNumericFormat(int)	Returns the OracleDbType value of the VECTOR column numeric format
GetVectorNumericFormat(string)	Returns the OracleDbType value of the VECTOR column numeric format
GetXmlReader	Returns the value of an XMLType column as an instance of an .NET XmlTextReader
IsDBNull	Indicates whether or not the column value is null
IsDBNullAsync	Returns a Task-based asynchronous version of OracleDataReader.IsDBNull() (Overloaded)
NextResult	Advances the data reader to the next result set when reading the results
NextResultAsync	Returns a Task-based asynchronous version of OracleDataReader.NextResult() (Overloaded)
Read	Reads the next row in the result set
ReadAsync	Returns a Task-based asynchronous version of OracleDataReader.Read() (Overloaded)
ToString	Inherited from System.Object

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)

OracleDataReader Static Methods

The OracleDataReader static method is listed in [Table 6-84](#).

Table 6-84 OracleDataReader Static Method

Method	Description
<code>Equals</code>	Inherited from <code>System.Object</code> (Overloaded)

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

OracleDataReader Properties

OracleDataReader properties are listed in [Table 6-85](#).

Table 6-85 OracleDataReader Properties

Property	Description
Depth	Gets a value indicating the depth of nesting for the current row
FetchSize	Specifies the size of OracleDataReader's internal cache
FieldCount	Gets the number of columns in the result set
HasRows	Indicates whether the OracleDataReader has one or more rows
HiddenFieldCount	Gets the number of fields in the OracleDataReader that are hidden
IsClosed	Indicates whether or not the data reader is closed
Item	Gets the value of the column (Overloaded)
InitialLOBFetchSize	Specifies the amount that the OracleDataReader initially fetches for LOB columns
InitialLONGFetchSize	Specifies the amount that the OracleDataReader initially fetches for LONG and LONG RAW columns
RecordsAffected	Gets the number of rows changed, inserted, or deleted by execution of the SQL statement
RowSize	Gets the amount of memory the internal cache of the OracleDataReader needs to store one row of data
RowsToFetchPerRoundTrip	Specifies the total number of rows to retrieve per database round trip <i>Not Available in ODP.NET, Unmanaged Driver</i>
SuppressGetDecimalInvalidCastException	Specifies whether to suppress the <code>InvalidCastException</code> and return a rounded-off 28 precision value if the Oracle NUMBER value has more than 28 precision.
UseEdmMapping	Indicates whether or not the OracleDataReader utilizes the Entity Data Model mapping configuration when returning values

Table 6-85 (Cont.) OracleDataReader Properties

Property	Description
VisibleFieldCount	Gets the number of fields in the OracleDataReader that are not hidden

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

Depth

This property gets a value indicating the depth of nesting for the current row.

Declaration

```
// C#  
public override int Depth {get;}
```

Property Value

The depth of nesting for the current row.

Implements

IDataReader

Exceptions

InvalidOperationException - The reader is closed.

Remarks

Default = 0

This property always returns zero because Oracle does not support nesting.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

FetchSize

This property specifies the size of OracleDataReader's internal cache.

Declaration

```
// C#  
public long FetchSize {get; set;}
```

Property Value

A long that specifies the amount of memory (in bytes) that the `OracleDataReader` uses for its internal cache.

Exceptions

`ArgumentException` - The `FetchSize` value specified is invalid.

Remarks

Default = The `OracleCommand`'s `FetchSize` property value.

The `FetchSize` property is inherited by the `OracleDataReader` that is created by a command execution returning a result set. The `FetchSize` property on the `OracleDataReader` object determines the amount of data fetched into its internal cache for each database round-trip.

The `RowSize` and `FetchSize` properties handle UDT and `XMLType` data differently than other scalar data types. Because only a reference to the UDT and `XMLType` data is stored in the ODP.NET's internal cache, the `RowSize` property accounts for only the memory needed for the reference (which is very small) and not the actual size of the UDT and `XMLType` data. Thus, applications can inadvertently fetch a large number of UDT or `XMLType` instances from the database in a single database round-trip. This is because the actual size of UDT and `XMLType` data does not count against the `FetchSize`, and it would require numerous UDT and `XMLType` references to fill up the default cache size of 131072 bytes. Therefore, when fetching UDT or `XMLType` data, the `FetchSize` property must be appropriately configured to control the number of UDT and `XMLType` instances that are to be fetched, rather than the amount of the actual UDT and `XMLType` data to be fetched.

NOTE: For LOB and LONG data types, only the sizes specified in the `InitialLOBFetchSize` and `InitialLONGFetchSize` properties are accounted for by the `RowSize` property in addition to the metadata and reference information that is maintained by the cache for each LOB in the select list.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)
- `OracleDataReader` "[RowSize](#)"
- `OracleCommand` "[ExecuteReader\(\)](#)"
- `OracleCommand` "[RowSize](#)"

FieldCount

This property returns the number of columns in the result set.

Declaration

```
// C#  
public override int FieldCount {get;}
```

Property Value

The number of columns in the result set if one exists, otherwise 0.

Implements

IDataRecord

Exceptions

InvalidOperationException - The reader is closed.

Remarks

Default = 0

This property has a value of 0 for queries that do not return result sets.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

HasRows

This property indicates whether the OracleDataReader has one or more rows.

Declaration

```
// C#  
public override bool HasRows {get;}
```

Return Value

bool

Remarks

HasRows indicates whether or not the OracleDataReader has any rows.

The value of HasRows does not change based on the row position. For example, even if the application has read all the rows from the result set and the next Read method invocation will return false, the HasRows property still returns true since the result set was not empty to begin with.

Rows are fetched to determine the emptiness of the OracleDataReader when HasRows property is accessed for the first time after the creation of the OracleDataReader object.

Example

```
// C#

using System;
using Oracle.DataAccess.Client;

class HasRowsSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleCommand cmd = new OracleCommand(
            "select * from emp where empno = 9999", con);

        OracleDataReader reader = cmd.ExecuteReader();

        if (!reader.HasRows)
            Console.WriteLine("The result set is empty.");
        else
            Console.WriteLine("The result set is not empty.");

        con.Dispose();
    }
}
```

📘 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)
- <http://msdn.microsoft.com/library> for detailed information about this Microsoft .NET Framework feature

HiddenFieldCount

This property gets the number of fields in the OracleDataReader that are hidden.

Declaration

```
// C#
public int HiddenFieldcount { get; }
```

Property Value

The number of fields in the OracleDataReader that are hidden.

Exceptions

InvalidOperationException - The reader is closed.

Remarks

OracleDataReader.FieldCount and OracleDataReader.VisibleFieldCount return the visible field count.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)
- ["VisibleFieldCount"](#)
- ["FieldCount "](#)

IsClosed

This property indicates whether or not the data reader is closed.

Declaration

```
// C#  
public override bool IsClosed {get;}
```

Property Value

If the OracleDataReader is in a closed state, returns true; otherwise, returns false.

Implements

IDataReader

Remarks

Default = true

IsClosed and RecordsAffected are the only two properties that are accessible after the OracleDataReader is closed.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

Item

This property gets the value of the column in .NET data type.

Overload List:

- [Item \[index\]](#)
This property gets the .NET Value of the column specified by the column index.
- [Item \[string\]](#)
This property gets the .NET Value of the column specified by the column name.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

Item [index]

This property gets the .NET Value of the column specified by the column index.

Declaration

```
// C#  
public override object this[int index] {get;}
```

Parameters

- *index*
The zero-based index of the column.

Property Value

The .NET value of the specified column.

Implements

IDataRecord

Remarks

Default = Not Applicable

In C#, this property is the indexer for this class.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

Item [string]

This property gets the .NET Value of the column specified by the column name.

Declaration

```
// C#  
public override object this[string columnName] {get;}
```

Parameters

- *columnName*
The name of the column.

Property Value

The .NET Value of the specified column.

Implements

IDataRecord

Remarks

Default = Not Applicable

A case-sensitive search is made to locate the specified column by its name. If this fails, then a case-insensitive search is made.

In C#, this property is the indexer for this class.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

InitialLOBFetchSize

This property specifies the amount that the OracleDataReader initially fetches for LOB columns.

Declaration

```
// C#  
public int InitialLOBFetchSize {get;}
```

Property Value

The size of the chunk to retrieve.

Exceptions

InvalidOperationException - The reader is closed.

Remarks

The maximum value supported for `InitialLOBFetchSize` is 2 GB.

Default is the `OracleCommand.InitialLOBFetchSize`, from which this value is inherited.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)
- ["InitialLOBFetchSize"](#) for further information on `OracleCommand.InitialLOBFetchSize`
- ["Obtaining LOB Data"](#)

InitialLONGFetchSize

This property specifies the amount that the `OracleDataReader` initially fetches for LONG and LONG RAW columns.

Declaration

```
// C#  
public long InitialLONGFetchSize {get;}
```

Property Value

The size of the chunk to retrieve. The default is 0.

Exceptions

`InvalidOperationException` - The reader is closed.

Remarks

The maximum value supported for `InitialLONGFetchSize` is 32767. If this property is set to a higher value, the provider resets it to 32767.

Default is `OracleCommand.InitialLONGFetchSize`, from which this value is inherited.

This property is read-only for the `OracleDataReader`.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)
- ["InitialLONGFetchSize"](#) for further information on `OracleCommand.InitialLONGFetchSize`
- ["Obtaining LONG and LONG RAW Data"](#)

RecordsAffected

This property gets the number of rows changed, inserted, or deleted by execution of the SQL statement.

Declaration

```
// C#  
public int RecordsAffected {get;}
```

Property Value

The number of rows affected by execution of the SQL statement.

Implements

`IDataReader`

Remarks

Default = 0

The value of -1 is returned for SELECT statements.

`IsClosed` and `RecordsAffected` are the only two properties that are accessible after the `OracleDataReader` is closed.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

RowSize

This property gets the amount of memory the internal cache of the `OracleDataReader` needs to store one row of data.

Declaration

```
// C#  
public long RowSize {get;}
```

Property Value

A long that indicates the amount of memory (in bytes) that an `OracleDataReader` needs to store one row of data for the executed query.

Remarks

The `RowSize` property is set to a nonzero value when the `OracleDataReader` object is created. This property can be used at design time or dynamically during runtime, to set the `FetchSize` property, based on the number of rows. For example, to enable the `OracleDataReader` object to fetch *N* rows for each database round-trip, the `OracleDataReader` `FetchSize` property can be set dynamically to `RowSize * N`. Note that for the `FetchSize` property to take effect appropriately, it must be set before the first invocation of `OracleDataReader.Read()` for the particular result set.

ODP.NET now supports values up to 32K for `VARCHAR2`, `NVARCHAR2` or `RAW` type columns in its calculation of `RowSize` value

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)
- ["FetchSize"](#)

RowsToFetchPerRoundTrip

This property specifies the total number of rows to retrieve per database round trip.

Declaration

```
// C#  
public Int64 RowsToFetchPerRoundTrip { get; set; }
```

Property Type

`System.Int64`

Remarks

This property has no default value. If it is not set, then ODP.NET ignores this property.

The row data fetched in a single round trip applies to scalar types only, such as `NUMBER` and `VARCHAR2` columns. If reference data types, such as LOBs, UDTs, and `XMLType`, exist in the result set, they are retrieved in separate round trips. These reference types can have their own fetch tuning parameters as reference type data sizes can vary in size from row to row, sometimes significantly.

RowsToFetchPerRoundTrip can be set before or after the `OracleCommand` executes on `OracleConnection`, `OracleCommand`, `OracleDataReader`, or `OracleRefCursor`. The value can also be changed after initial data fetches so that more or fewer rows are fetched on subsequent round trips. For `OracleConfiguration`, this property can only be set before the first connection opens.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

SuppressGetDecimalInvalidCastException

This property specifies whether to suppress the `InvalidCastException` and return a rounded-off 28 precision value if the Oracle `NUMBER` value has more than 28 precision.

Declaration

```
// C#  
public bool SuppressGetDecimalInvalidCastException { get; set; }
```

Property Type

`System.Boolean`

Remarks

Oracle `NUMBER` has a maximum of 38 precision. .NET `Decimal` has a maximum of 28 or 29 precision. When the `GetDecimal()` method is called for an Oracle `NUMBER` value that cannot be represented as a .NET `Decimal`, then ODP.NET throws an `InvalidCastException` because not all the precision can be preserved when converting the number to a .NET `Decimal`.

This behavior occurs when `SuppressGetDecimalInvalidCastException` is set to `false`. Its default value is `false`.

When `SuppressGetDecimalInvalidCastException` is set to `true`, then the resulting decimal will be rounded off to 28 or 29 precision so that it can fit as a .NET decimal without throwing an exception. If the resulting rounded number is larger than can be stored in a .NET `Decimal`, an exception will be thrown, such as the number 1×10^{32} .

The `SuppressGetDecimalInvalidCastException` property can also be configured on `OracleConfiguration`, `OracleConnection`, and `OracleDataAdapter` objects. The `OracleDataReader SuppressGetDecimalInvalidCastException` property value inherits `OracleConnection SuppressGetDecimalInvalidCastException` property value at construction time.

By using `OracleDataReader.SuppressGetDecimalInvalidCastException`, this ODP.NET setting becomes enabled for reader's lifetime.

See Also

- OracleConfiguration [SuppressGetDecimalInvalidCastException](#)
- OracleConnection [SuppressGetDecimalInvalidCastException](#)
- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

UseEdmMapping

This read-only property indicates whether or not the OracleDataReader utilizes the Entity Data Model mapping configuration when returning values.

Declaration

```
// C#  
public bool UseEdmMapping {get;}
```

Property Value

A boolean that indicates whether the OracleDataReader uses the Entity Data Model mapping configuration for returning values.

Remarks

Default is false.

The value is inherited from the OracleCommand object.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)
- [OracleCommand Class](#)

VisibleFieldCount

This property gets the number of fields in the OracleDataReader that are not hidden.

Declaration

```
// C#  
public override int VisibleFieldcount { get; }
```

Property Value

The number of fields that are not hidden.

Exceptions

InvalidOperationException - The reader is closed.

Remarks

If an application sets the AddRowid property on an OracleCommand object to true, then the application can access the RowId but it is not a visible field. If RowId is added in the select statement list, then it is a visible field. OracleDataReader.VisibleFieldCount and OracleDataReader.FieldCount always have the same value.

Example

```
// C#

using System;
using System.Data;
using System.Data.Common;
using Oracle.DataAccess.Client;

class VisibleFieldCountSample
{
    static void Main(string[] args)
    {
        string constr = "User Id=scott; Password=tiger; Data Source=oracle;";
        DbProviderFactory factory =
            DbProviderFactories.GetFactory("Oracle.DataAccess.Client");

        using (DbConnection conn = factory.CreateConnection())
        {
            conn.ConnectionString = constr;
            try
            {
                conn.Open();
                OracleCommand cmd = (OracleCommand)factory.CreateCommand();
                cmd.Connection = (OracleConnection)conn;

                //to gain access to ROWIDs of the table
                cmd.AddRowid = true;
                cmd.CommandText = "select empno, ename from emp;";

                OracleDataReader reader = cmd.ExecuteReader();

                int visFC = reader.VisibleFieldCount; //Results in 2
                int hidFC = reader.HiddenFieldCount; // Results in 1

                Console.WriteLine("Visible field count: " + visFC);
                Console.WriteLine("Hidden field count: " + hidFC);

                reader.Dispose();
                cmd.Dispose();
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
                Console.WriteLine(ex.StackTrace);
            }
        }
    }
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)
- ["VisibleFieldCount"](#)
- ["FieldCount "](#)

OracleDataReader Public Methods

OracleDataReader public methods are listed in [Table 6-86](#).

Table 6-86 OracleDataReader Public Methods

Public Method	Description
Close	Closes the OracleDataReader
CreateObjRef	Inherited from System.MarshalByRefObject
Dispose	Releases any resources or memory allocated by the object
Equals	Inherited from System.Object (Overloaded)
GetBoolean	Returns the bool value of the specified numeric column
GetByte	Returns the byte value of the specified column
GetByteArray(int)	Returns the byte array value of the specified numeric column <i>Not Available in ODP.NET, Unmanaged Driver</i>
GetByteArray(string)	Returns the byte array value of the specified numeric column <i>Not Available in ODP.NET, Unmanaged Driver</i>
GetBytes	Populates the provided byte array with up to the maximum number of bytes, from the specified offset (in bytes) of the column
GetChar	<i>Not Supported</i>
GetChars	Populates the provided character array with up to the maximum number of characters, from the specified offset (in characters) of the column
GetData	<i>Not Supported</i>
GetDataTypeName	Returns the ODP.NET type name of the specified column
GetDateTime	Returns the DateTime value of the specified column
GetDateTimeOffset	Returns the DateTimeOffset value of the specified column
GetDecimal	Returns the decimal value of the specified NUMBER column
GetDouble	Returns the double value of the specified NUMBER column or BINARY_DOUBLE column
GetDoubleArray(int)	Returns the double[] value of the specified VECTOR column
GetDoubleArray(string)	Returns the double[] value of the specified VECTOR column

Table 6-86 (Cont.) OracleDataReader Public Methods

Public Method	Description
GetEnumerator	Returns an IEnumerator that can be used to iterate through the collection
GetFieldType	Returns the Type of the specified column
GetFieldValueAsync	Returns a Task-based asynchronous version of <code>OracleDataReader.GetFieldValue()</code> , which returns the column value as a specified type (Overloaded)
GetFloat	Returns the <code>float</code> value of the specified NUMBER column or BINARY_FLOAT column
GetFloatArray(int)	Returns the <code>float[]</code> value of the specified VECTOR column
GetFloatArray(string)	Returns the <code>float[]</code> value of the specified VECTOR column
GetGuid	Returns the Guid value of the specified binary data column
<code>GetHashCode</code>	Inherited from <code>System.Object</code>
GetInt16	Returns the <code>Int16</code> value of the specified NUMBER column
GetInt16Array(int)	Returns the <code>short[]</code> value of the specified VECTOR column
GetInt16Array(string)	Returns the <code>short[]</code> value of the specified VECTOR column
GetInt32	Returns the <code>Int32</code> value of the specified NUMBER column
GetInt64	Returns the <code>Int64</code> value of the specified NUMBER column
<code>GetLifetimeService</code>	Inherited by <code>System.MarshalByRefObject</code>
GetName	Returns the name of the specified column
GetNonZeroDoubleArray	Returns the non-zero-dimension values <code>double[]</code> of the specified sparse VECTOR column.(Overloaded) <i>Not Available in ODP.NET, Unmanaged Driver</i>
GetNonZeroFloatArray	Returns the non-zero-dimension values <code>float[]</code> of the specified sparse VECTOR column.(Overloaded) <i>Not Available in ODP.NET, Unmanaged Driver</i>
GetNonZeroIndexArray	Returns the non-zero-dimension indices <code>long[]</code> of the specified sparse VECTOR column.(Overloaded) <i>Not Available in ODP.NET, Unmanaged Driver</i>
GetNonZeroInt16Array	Returns the non-zero-dimension values <code>short[]</code> of the specified sparse VECTOR column. (Overloaded) <i>Not Available in ODP.NET, Unmanaged Driver</i>
GetOracleBFile	Returns an <code>OracleBFile</code> object of the specified BFILE column
GetOracleBinary	Returns an <code>OracleBinary</code> structure of the specified column
GetOracleBlob	Returns an <code>OracleBlob</code> object of the specified BLOB column
GetOracleBlobForUpdate	Returns an updatable <code>OracleBlob</code> object of the specified BLOB column <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
GetOracleBoolean(int)	Returns the <code>OracleBoolean</code> value of the column
GetOracleClob	Returns an <code>OracleClob</code> object of the specified CLOB column

Table 6-86 (Cont.) OracleDataReader Public Methods

Public Method	Description
GetOracleClobForUpdate	Returns an updatable <code>OracleClob</code> object of the specified CLOB column <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
GetOracleDataReader	Returns the <code>OracleDataReader</code> object corresponding to the REF CURSOR type column in the select list <i>Not Available in ODP.NET, Unmanaged Driver</i>
GetOracleDate	Returns an <code>OracleDate</code> structure of the specified DATE column
GetOracleDecimal	Returns an <code>OracleDecimal</code> structure of the specified NUMBER column
GetOracleIntervalDS	Returns an <code>OracleIntervalDS</code> structure of the specified INTERVAL DAY TO SECOND column
GetOracleIntervalYM	Returns an <code>OracleIntervalYM</code> structure of the specified INTERVAL YEAR TO MONTH column
GetOracleRef	Returns an <code>OracleRef</code> object of the specified REF column
GetOracleRefCursor	Returns the <code>OracleRefCursor</code> object corresponding to the REF CURSOR type column in the select list <i>Not Available in ODP.NET, Unmanaged Driver</i>
GetOracleString	Returns an <code>OracleString</code> structure of the specified column
GetOracleTimeStamp	Returns an <code>OracleTimeStamp</code> structure of the Oracle TimeStamp column
GetOracleTimeStampLTZ	Returns an <code>OracleTimeStampLTZ</code> structure of the specified Oracle TimeStamp WITH LOCAL TIME ZONE column
GetOracleTimeStampTZ	Returns an <code>OracleTimeStampTZ</code> structure of the specified Oracle TimeStamp WITH TIME ZONE column
GetOracleXmlType	Returns an <code>OracleXmlType</code> object of the specified XMLType column
GetOracleValue	Returns the specified column value as a ODP.NET type
GetOracleValues	Gets all the column values as ODP.NET types
GetOracleVector	Returns <code>OracleVector</code> object of the specified VECTOR column. (Overloaded) <i>Not Available in ODP.NET, Unmanaged Driver</i>
GetOrdinal	Returns the 0-based ordinal (or index) of the specified column name
GetProviderSpecificFieldType	Returns the provider-specific type of the specified column
GetProviderSpecificValue	Returns an object that represents the underlying provider-specific value of the specified ordinal
GetProviderSpecificValues	Returns an array of objects that represent the underlying provider-specific values
GetSchemaTable	Returns a <code>DataTable</code> that describes the column metadata of the <code>OracleDataReader</code>
GetString	Returns the string value of the specified column
GetTimeSpan	Returns the <code>TimeSpan</code> value of the specified INTERVAL DAY TO SECOND column

Table 6-86 (Cont.) OracleDataReader Public Methods

Public Method	Description
GetType	Inherited from System.Object class
GetValue	Returns the column value as a .NET type
GetValues	Gets all the column values as .NET types
GetVectorDimensions	Returns the number of dimensions of the specified VECTOR column.(Overloaded) <i>Not Available in ODP.NET, Unmanaged Driver</i>
GetVectorNumericFormat(int)	Returns the OracleDbType value of the VECTOR column numeric format
GetVectorNumericFormat(string)	Returns the OracleDbType value of the VECTOR column numeric format
GetXmlReader	Returns the value of an XMLType column as an instance of an .NET XmlTextReader
IsDBNull	Indicates whether or not the column value is null
IsDBNullAsync	Returns a Task-based asynchronous version of OracleDataReader.IsDBNull() (Overloaded)
NextResult	Advances the data reader to the next result set when reading the results
NextResultAsync	Returns a Task-based asynchronous version of OracleDataReader.NextResult() (Overloaded)
Read	Reads the next row in the result set
ReadAsync	Returns a Task-based asynchronous version of OracleDataReader.Read() (Overloaded)
ToString	Inherited from System.Object

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

Close

This method closes the OracleDataReader.

Declaration

```
// C#
public override void Close();
```

Implements

IDataReader

Remarks

The Close method frees all resources associated with the OracleDataReader.

Example

The code example for the OracleDataReader class includes the Close method. See OracleDataReader Overview "[Example](#)".

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

Dispose

This method releases any resources or memory allocated by the object.

Declaration

```
// C#  
public void Dispose();
```

Implements

IDisposable

Remarks

The Dispose method also closes the OracleDataReader.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetBoolean

This method returns the bool value of the specified numeric column.

Declaration

```
// C#  
public override bool GetBoolean(int index);
```

Parameters

- *index*

The zero-based column index.

Return Value

The bool value of the column.

Implements

IDataRecord

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type.

Remarks

- This method can be called on the `BOOLEAN` column, any numeric database type column (`NUMBER`, `BINARY_DOUBLE`, `BINARY_FLOAT`, and `FLOAT`), or any scalar character based column (`CHAR`, `NCHAR`, `VARCHAR2`, `NVARCHAR2`).
- For numeric columns, `FALSE` is returned if column value is 0; `TRUE` otherwise.
- For scalar character based column (`CHAR`, `NCHAR`, `VARCHAR2`, `NVARCHAR2`), `GetBoolean` can be called on Valid String literals.

Valid String literals to represent `TRUE` and `FALSE` (case insensitive) are as follows:

- For `TRUE`, valid string literals are (case insensitive) - `true`, `yes`, `on`, `1`, `t`, `y`
- For `FALSE`, valid string literals are (case insensitive) - `false`, `no`, `off`, `0`, `f`, `n`

If the string literals are anything other than those listed above, then an `InvalidCastException` will be returned when trying to read the value.

- `IsDBNull` should be called to check for `NULL` values before calling this method.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetByte

This method returns the byte value of the specified column.

Declaration

```
// C#  
public override byte GetByte(int index);
```

Parameters

- *index*

The zero-based column index.

Return Value

The value of the column as a byte.

Implements

IDataRecord

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

IsDBNull should be called to check for NULL values before calling this method.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetByteArray(int)

This method returns the byte array value of the specified VECTOR column.

Declaration

```
// C#  
public override byte[] GetByteArray(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The byte array value of the column. A sparse vector type column will return the array value as a dense vector.

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

While this accessor can retrieve data from columns of any numeric format, it is the only accessor for retrieving `byte[]` for `BINARY` format columns.

`IsDBNull` should be called to check for `NULL` values before calling this method.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetByteArray(string)

This method returns the byte array value of the specified `VECTOR` column.

Declaration

```
// C#  
public override byte[] GetByteArray(string name);
```

Parameters

- *name*
The specified column name

Return Value

The byte array value of the column. A sparse vector type column will return the array value as a dense vector.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

While this accessor can retrieve data from columns of any numeric format, it is the only accessor for retrieving `byte[]` for `BINARY` format columns.

`IsDBNull` should be called to check for `NULL` values before calling this method.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetBytes

This method populates the provided byte array with up to the maximum number of bytes, from the specified offset (in bytes) of the column.

Declaration

```
// C#  
public override long GetBytes(int index, long fieldOffset, byte[] buffer,  
    int bufferOffset, int length);
```

Parameters

- *index*
The zero-based column index.
- *fieldOffset*
The offset within the column from which reading begins (in bytes).
- *buffer*
The byte array that the data is read into.
- *bufferOffset*
The offset within the buffer to begin reading data into (in bytes).
- *length*
The maximum number of bytes to read (in bytes).

Return Value

The number of bytes read.

Implements

IDataRecord

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

This method returns the number of bytes read into the buffer. This may be less than the actual length of the field if the method has been called previously for the same column.

If a null reference is passed for buffer, the length of the field in bytes is returned.

`IsDBNull` should be called to check for NULL values before calling this method.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetChar

This method is not supported.

Declaration

```
// C#  
public override long GetChar(int index);
```

Parameters

- *index*
The zero based column index.

Implements

`IDataRecord`

Exceptions

`NotSupportedException` - This property is not supported.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetChars

This method populates the provided character array with up to the maximum number of characters, from the specified offset (in characters) of the column.

Declaration

```
// C#  
public override long GetChars(int index, long fieldOffset, char[] buffer,  
    int bufferOffset, int length);
```

Parameters

- *index*
The zero based column index.
- *fieldOffset*
The index within the column from which to begin reading (in characters).
- *buffer*
The character array that the data is read into.
- *bufferOffset*
The index within the buffer to begin reading data into (in characters).
- *length*
The maximum number of characters to read (in characters).

Return Value

The number of characters read.

Implements

IDataRecord

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, *Read()* has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

This method returns the number of characters read into the buffer. This may be less than the actual length of the field, if the method has been called previously for the same column.

If a null reference is passed for *buffer*, the length of the field in characters is returned.

IsDBNull should be called to check for NULL values before calling this method.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetData

This method is not supported

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetDataTypeName

This method returns the ODP.NET type name of the specified column.

Declaration

```
// C#  
public override string GetDataTypeName(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The name of the ODP.NET type of the column.

Implements

IDataRecord

Exceptions

InvalidOperationException - The reader is closed.
IndexOutOfRangeException - The column index is invalid.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetDateTime

This method returns the `DateTime` value of the specified column.

Declaration

```
// C#  
public override DateTime GetDateTime(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The `DateTime` value of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetDateTimeOffset

This method returns the `DateTimeOffset` value of a `TIMESTAMP WITH TIME ZONE` column. `DateTimeOffset` represents a point in time. It is generally expressed as a date and time relative to Coordinated Universal Time (UTC).

Declaration

```
// C#  
public override DateTimeOffset GetDateTimeOffset(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The `DateTimeOffset` value of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

- This method can be called only on a `TIMESTAMP WITH TIME ZONE` column.
- `IsDBNull` should be called to check for `NULL` values before calling this method.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetDecimal

This method returns the decimal value of the specified `NUMBER` column.

Declaration

```
// C#  
public override decimal GetDecimal(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The decimal value of the column.

Implements

IDataRecord

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

IsDBNull should be called to check for NULL values before calling this method.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetDouble

This method returns the double value of the specified NUMBER column or BINARY_DOUBLE column.

Declaration

```
// C#  
public override double GetDouble(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The double value of the column.

Implements

IDataRecord

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

IsDBNull should be called to check for NULL values before calling this method.

GetDouble now supports retrieval of data from BINARY_DOUBLE columns.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetDoubleArray(int)

This method returns the double[] value of the specified VECTOR column.

Declaration

```
// C#  
public double[] GetDoubleArray(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The double[] value of the column. A sparse vector type column will return the array value as a dense vector.

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

IsDBNull should be called to check for NULL values before calling this method. This method cannot be called on BINARY format vector columns.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetDoubleArray(string)

This method returns the double[] value of the specified VECTOR column.

Declaration

```
// C#  
public double[] GetDoubleArray(string name);
```

Parameters

- *name*
The specified column name.

Return Value

The double[] value of the column. A sparse vector type column will return the array value as a dense vector.

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - Unable to find specified column in result set.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

IsDBNull should be called to check for NULL values before calling this method. This method cannot be called on BINARY format vector columns.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetEnumerator

This method returns an IEnumerator that can be used to iterate through the collection (record set).

Declaration

```
// C#  
public override IEnumerator GetEnumerator();
```

Return Value

An IEnumerator that can be used to iterate through the collection (record set).

Exceptions

InvalidOperationException - The reader is closed.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetFieldType

This method returns the type of the specified column.

Declaration

```
// C#  
public override Type GetFieldType(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The type of the default .NET type of the column.

Implements

IDataRecord

Exceptions

InvalidOperationException - The reader is closed, or the specified column is a UDT but no registered custom type mapping exists for the UDT.

IndexOutOfRangeException - The column index is invalid.

Remarks

GetFieldType returns a type that corresponds to the value that the application obtains after invoking the GetValue accessor or Item property on the OracleDataReader. For example, if the column is a string, this method returns a .NET Type object for a .NET string.

If the attribute is a UDT, this method may return either of the following:

- A .NET Type of the custom type if a custom type mapping exists for the Oracle object or collection.
- A .NET Type of string if the column is an Oracle REF.

For VECTOR columns, this method returns a numeric array or byte array type based on the VECTOR numeric format. BINARY vectors can only return a .NET byte[]. If the VECTOR numeric format is '*', then this method returns type of Array. For sparse VECTOR columns, this method returns type of string.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetFieldValueAsync

GetFieldValueAsync returns a Task-based asynchronous version of OracleDataReader.GetFieldValue(), which returns the column value as a specified type.

Overload List:

- This method returns a Task-based asynchronous version of OracleDataReader.GetFieldValue()
- This method returns a Task-based asynchronous version of OracleDataReader.GetFieldValue()

GetFieldValueAsync(int, CancellationToken)

Declaration

```
// C#  
public override Task<T> GetFieldValueAsync<T>(int ordinal, CancellationToken  
cancellationTokentoken);
```

Type Parameters

T - The type of the value to be returned.

Parameters

ordinal - The zero-based column index.

`cancellationToken` - The input cancellation token which can be used by the application to cancel the task.

Return Value

`Task<T>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Implements

`DbDataReader`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` – The value returned by the database cannot be cast to `T`.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetFieldValueAsync(int)

Declaration

```
// C#  
public Task<T> GetFieldValueAsync<T>(int ordinal);
```

Type Parameters

`T` - The type of the value to be returned.

Parameters

`ordinal` - The zero-based column index.

Return Value

`Task<T>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Implements

`DbDataReader`

Remarks

This will call into the `GetFieldValueAsync` implementation with argument `cancellationToken` passed as `CancellationToken.None`.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` – The value returned by the database cannot be cast to `T`.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetFloat

This method returns the float value of the specified `NUMBER` column or `BINARY_FLOAT` column.

Declaration

```
// C#  
public override float GetFloat(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The float value of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

`GetFloat` now supports retrieval of data from `BINARY_FLOAT` columns.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetFloatArray(int)

This method returns the `float[]` value of the specified `VECTOR` column.

Declaration

```
// C#  
public float[] GetFloatArray(int index);
```

Parameters

- *index*

The zero-based column index.

Return Value

The `float[]` value of the column. A sparse vector type column will return the array value as a dense vector.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method. This method cannot be called on `BINARY` format vector columns.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetFloatArray(string)

This method returns the `float[]` value of the specified `VECTOR` column.

Declaration

```
// C#  
public float[] GetFloatArray(string name);
```

Parameters

- *name*

The specified column name.

Return Value

The float[] value of the column. A sparse vector type column will return the array value as a dense vector.

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - Unable to find specified column in result set.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

IsDBNull should be called to check for NULL values before calling this method. This method cannot be called on BINARY format vector columns.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetGuid

This method returns the Guid value of the specified binary data column. A GUID is a Globally Unique Identifier.

Declaration

```
// C#  
public override Guid GetGuid(int index);
```

Parameters

- *index*

The zero-based column index.

Return Value

The Guid value of the column.

Implements

IDataRecord

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

System.ArgumentException - Byte array for GUID must be exactly 16 bytes long.

Remarks

- This method can be called on columns that contain binary data, such as RAW and BLOB.
- An exception will be thrown if the binary data of the column is not 16 bytes in length.
- IsDBNull should be called to check for NULL values before calling this method.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetInt16

This method returns the Int16 value of the specified NUMBER column.

Note

short is equivalent to Int16.

Declaration

```
// C#  
public override short GetInt16(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The Int16 value of the column.

Implements

IDataRecord

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

IsDBNull should be called to check for NULL values before calling this method.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetInt16Array(int)

This method returns the short[] value of the specified VECTOR column.

Declaration

```
// C#  
public short[] GetInt16Array(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The short[] value of the column. A sparse vector type column will return the array value as a dense vector.

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

IsDBNull should be called to check for NULL values before calling this method. This method cannot be called on BINARY format vector columns.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetInt16Array(string)

This method returns the short[] value of the specified VECTOR column.

Declaration

```
// C#  
public short[] GetInt16Array(string name);
```

Parameters

- *name*
The specified column name.

Return Value

The short[] value of the column. A sparse vector type column will return the array value as a dense vector.

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - Unable to find specified column in result set.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

IsDBNull should be called to check for NULL values before calling this method. This method cannot be called on BINARY format vector columns.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetInt32

This method returns the Int32 value of the specified NUMBER column.

Note

int is equivalent to Int32.

Declaration

```
// C#  
public override int GetInt32(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The Int32 value of the column.

Implements

IDataRecord

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

IsDBNull should be called to check for NULL values before calling this method.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetInt64

This method returns the Int64 value of the specified NUMBER column.

Note

long is equivalent to Int64.

Declaration

```
// C#  
public override long GetInt64(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The Int64 value of the column.

Implements

IDataRecord

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

IsDBNull should be called to check for NULL values before calling this method.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetName

This method returns the name of the specified column.

Declaration

```
// C#  
public override string GetName(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The name of the column.

Implements

IDataRecord

Exceptions

InvalidOperationException - The reader is closed.

IndexOutOfRangeException - The column index is invalid.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetNonZeroDoubleArray

This method returns the non-zero-dimension values `double[]` of the specified sparse VECTOR column.

Overload List:

- [GetNonZeroDoubleArray\(int\)](#)
This method returns the non-zero-dimension values `double[]` of the specified sparse VECTOR column.
- [GetNonZeroDoubleArray\(string\)](#)
This method returns the non-zero-dimension values `double[]` of the specified sparse VECTOR column.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetNonZeroDoubleArray(int)

This method returns the non-zero-dimension values `double[]` of the specified sparse VECTOR column.

Declaration

```
// C#  
public double[] GetNonZeroDoubleArray(int index);
```

Parameters

index

A zero-based column index.

Return Value

The non-zero-dimension values `double[]` of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type, the VECTOR column is a dense vector type column, a BINARY format vector column, or the column value is NULL.

Remarks

`IsDBNull` should be called to check for NULL values before calling this method.

Only call this method on a sparse VECTOR column where `SchemaTable["VectorType"]` is `OracleVectorType.Sparse`. This method cannot be called on BINARY format vector columns.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetNonZeroDoubleArray(string)

This method returns the non-zero-dimension values `double[]` of the specified sparse VECTOR column.

Declaration

```
// C#  
public double[] GetNonZeroDoubleArray(string name);
```

Parameters

name

The specified column name.

Return Value

The non-zero-dimension values `double[]` of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called or all rows have been read.

`IndexOutOfRangeException` - Unable to find specified column in result set.

`InvalidCastException` - The accessor method is invalid for this column type, the `VECTOR` column is a dense vector type column, a `BINARY` format vector column, or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

Only call this method on a sparse `VECTOR` column where `SchemaTable["VectorType"]` is `OracleVectorType.Sparse`. This method cannot be called on `BINARY` format vector columns.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetNonZeroFloatArray

This method returns the non-zero-dimension values `float[]` of the specified sparse `VECTOR` column.

Overload List:

- [GetNonZeroFloatArray\(int\)](#)
This method returns the non-zero-dimension indices `float[]` of the specified sparse `VECTOR` column.
- [GetNonZeroFloatArray\(string\)](#)
This method returns the non-zero-dimension values `float[]` of the specified sparse `VECTOR` column.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetNonZeroFloatArray(int)

This method returns the non-zero-dimension values `float[]` of the specified sparse VECTOR column.

Declaration

```
// C#  
public float[] GetNonZeroFloatArray(int index);
```

Parameters

index

A zero-based column index.

Return Value

The non-zero-dimension values `float[]` of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type, the VECTOR column is a dense vector type column, a BINARY format vector column, or the column value is NULL.

Remarks

`IsDBNull` should be called to check for NULL values before calling this method.

Only call this method on a sparse VECTOR column where `SchemaTable["VectorType"]` is `OracleVectorType.Sparse`. This method cannot be called on BINARY format vector columns.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetNonZeroFloatArray(string)

This method returns the non-zero-dimension values `float[]` of the specified sparse VECTOR column.

Declaration

```
// C#  
public float[] GetNonZeroFloatArray(string name);
```

Parameters

name

The specified column name.

Return Value

The non-zero-dimension values `float[]` of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called or all rows have been read.

`IndexOutOfRangeException` - Unable to find specified column in result set.

`InvalidCastException` - The accessor method is invalid for this column type, the VECTOR column is a dense vector type column, a BINARY format vector column, or the column value is NULL.

Remarks

`IsDBNull` should be called to check for NULL values before calling this method.

Only call this method on a sparse VECTOR column where `SchemaTable["VectorType"]` is `OracleVectorType.Sparse`. This method cannot be called on BINARY format vector columns.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetNonZeroIndexArray

This method returns the non-zero-dimension indices `long[]` of the specified sparse VECTOR column.

Overload List:

- [GetNonZeroIndexArray\(int\)](#)

This method returns the non-zero-dimension indices `long[]` of the specified sparse VECTOR column.

- [GetNonZeroIndexArray\(string\)](#)

This method returns the non-zero-dimension indices `long[]` of the specified sparse VECTOR column.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetNonZeroIndexArray(int)

This method returns the non-zero-dimension indices `long[]` of the specified sparse VECTOR column.

Declaration

```
// C#  
public long[] GetNonZeroIndexArray(int index);
```

Parameters

index

A zero-based column index.

Return Value

The non-zero-dimension indices `long[]` of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type, the VECTOR column is a dense vector type column, or the column value is NULL.

Remarks

`IsDBNull` should be called to check for NULL values before calling this method.

Only call this method on a sparse VECTOR column where `SchemaTable["VectorType"]` is `OracleVectorType.Sparse`.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetNonZeroIndexArray(string)

This method returns the non-zero-dimension indices `long[]` of the specified sparse VECTOR column.

Declaration

```
// C#  
public long[] GetNonZeroIndexArray(string name);
```

Parameters

name

The specified column name.

Return Value

The 1-based non-zero-dimension indices `long[]` of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called or all rows have been read.

`IndexOutOfRangeException` - Unable to find specified column in result set.

`InvalidCastException` - The accessor method is invalid for this column type, the VECTOR column is a dense vector type column, or the column value is NULL.

Remarks

`IsDBNull` should be called to check for NULL values before calling this method.

Only call this method on a sparse VECTOR column where `SchemaTable["VectorType"]` is `OracleVectorType.Sparse`.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetNonZeroInt16Array

This method returns the non-zero-dimension values `short[]` of the specified sparse VECTOR column.

Overload List:

- [GetNonZeroInt16Array\(int\)](#)

This method returns the non-zero-dimension indices `short[]` of the specified sparse VECTOR column.

- [GetNonZeroInt16Array\(string\)](#)

This method returns the non-zero-dimension values `short[]` of the specified sparse VECTOR column.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetNonZeroInt16Array(int)

This method returns the non-zero-dimension values `short[]` of the specified sparse VECTOR column.

Declaration

```
// C#  
public short[] GetNonZeroInt16Array(int index);
```

Parameters

index

A zero-based column index.

Return Value

The non-zero-dimension values `short[]` of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type, the VECTOR column is a dense vector type column, a BINARY format vector column, or the column value is NULL.

Remarks

`IsDBNull` should be called to check for NULL values before calling this method.

Only call this method on a sparse VECTOR column where `SchemaTable["VectorType"]` is `OracleVectorType.Sparse`. This method cannot be called on BINARY format vector columns.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetNonZeroInt16Array(string)

This method returns the non-zero-dimension values `short[]` of the specified sparse VECTOR column.

Declaration

```
// C#  
public short[] GetNonZeroInt16Array(string name);
```

Parameters

name

The specified column name.

Return Value

The non-zero-dimension values `short[]` of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called or all rows have been read.

`IndexOutOfRangeException` - Unable to find specified column in result set.

`InvalidCastException` - The accessor method is invalid for this column type, the VECTOR column is a dense vector type column, a BINARY format vector column, or the column value is NULL.

Remarks

`IsDBNull` should be called to check for NULL values before calling this method.

Only call this method on a sparse VECTOR column where `SchemaTable["VectorType"]` is `OracleVectorType.Sparse`. This method cannot be called on BINARY format vector columns.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetOracleBFile

This method returns an `OracleBFile` object of the specified BFILE column.

Declaration

```
// C#  
public OracleBFile GetOracleBFile(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The `OracleBFile` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetOracleBinary

This method returns an `OracleBinary` structure of the specified column.

Declaration

```
// C#  
public OracleBinary GetOracleBinary(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The `OracleBinary` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

IsDBNull should be called to check for NULL values before calling this method.

GetOracleBinary is used on the following Oracle types:

- BFILE
- BLOB
- LONG RAW
- RAW

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetOracleBlob

This method returns an OracleBlob object of the specified BLOB column.

Declaration

```
// C#  
public OracleBlob GetOracleBlob(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The OracleBlob value of the column.

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

IsDBNull should be called to check for NULL values before calling this method.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetOracleBlobForUpdate

GetOracleBlobForUpdate returns an updatable OracleBlob object of the specified BLOB column.

Overload List:

- [GetOracleBlobForUpdate\(int\)](#)
This method returns an updatable OracleBlob object of the specified BLOB column.
- [GetOracleBlobForUpdate\(int, int\)](#)
This method returns an updatable OracleBlob object of the specified BLOB column using a WAIT clause.

GetOracleBlobForUpdate(int)

This method returns an updatable OracleBlob object of the specified BLOB column.

Declaration

```
// C#  
public OracleBlob GetOracleBlobForUpdate(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

An updatable OracleBlob object.

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

When the OracleCommand's ExecuteReader() method is invoked, all the data fetched by the OracleDataReader is from a particular snapshot. Therefore, calling an accessor method on the same column always returns the same value. However, the GetOracleBlobForUpdate()

method incurs a database round-trip to obtain a reference to the current BLOB data while also locking the row using the FOR UPDATE clause. This means that the `OracleBlob` obtained from `GetOracleBlob()` can have a different value than the `OracleBlob` obtained from `GetOracleBlobForUpdate()` since it is not obtained from the original snapshot.

The returned `OracleBlob` object can be used to safely update the BLOB because the BLOB column has been locked after a call to this method.

Invoking this method internally executes a `SELECT . . FOR UPDATE` statement without a `WAIT` clause. Therefore, the statement can wait indefinitely until a lock is acquired for that row.

`IsDBNull` should be called to check for NULL values before calling this method.

Example

The following example gets the `OracleBlob` object for update from the reader, updates the `OracleBlob` object, and then commits the transaction.

```
/* Database Setup, if you have not done so yet.
connect scott/tiger@oracle
CREATE TABLE empInfo (
empno NUMBER(4) PRIMARY KEY,
empName VARCHAR2(20) NOT NULL,
hiredate DATE,
salary NUMBER(7,2),
jobDescription Clob,
byteCodes BLOB
);

Insert into empInfo(EMPNO,EMPNAME,JOBDESCRIPTION,byteCodes) values
(1,'KING','SOFTWARE ENGR', '5657');
Insert into empInfo(EMPNO,EMPNAME,JOBDESCRIPTION,byteCodes) values
(2,'SCOTT','MANAGER', '5960');
commit;

*/
// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class GetOracleBlobForUpdateSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        // Get the ByteCodes for empno = 1
        string cmdstr = "SELECT BYTECODES, EMPNO FROM EMPINFO where EMPNO = 1";
        OracleCommand cmd = new OracleCommand(cmdstr, con);

        // Since we are going to update the OracleBlob object, we will
        //have to create a transaction
        OracleTransaction txn = con.BeginTransaction();

        // Get the reader
        OracleDataReader reader = cmd.ExecuteReader();
```

```
// Declare the variables to retrieve the data in EmpInfo
OracleBlob byteCodesBlob;

// Read the first row
reader.Read();
if (!reader.IsDBNull(0))
{
    byteCodesBlob = reader.GetOracleBlobForUpdate(0);

    // Close the reader
    reader.Close();

    // Update the ByteCodes object
    byte[] addedBytes = new byte[2] {0, 0};
    byteCodesBlob.Append(addedBytes, 0, addedBytes.Length);

    // Now commit the transaction
    txn.Commit();
    Console.WriteLine("Blob Column successfully updated");
}
else
    reader.Dispose();

// Close the connection
con.Dispose();
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)
- ["LOB Support"](#)

GetOracleBlobForUpdate(int, int)

This method returns an updatable `OracleBlob` object of the specified BLOB column using a `WAIT` clause.

Declaration

```
// C#
public OracleBlob GetOracleBlobForUpdate(int index, int wait);
```

Parameters

- *index*
The zero-based column index.
- *wait*
The number of seconds the method waits to acquire a lock.

Return Value

An updatable `OracleBlob` object.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

When the `OracleCommand`'s `ExecuteReader()` method is invoked, all the data fetched by the `OracleDataReader` is from a particular snapshot. Therefore, calling an accessor method on the same column always returns the same value. However, the `GetOracleBlobForUpdate()` method incurs a database round-trip to obtain a reference to the current BLOB data while also locking the row using the `FOR UPDATE` clause. This means that the `OracleBlob` obtained from `GetOracleBlob()` can have a different value than the `OracleBlob` obtained from `GetOracleBlobForUpdate()` since it is not obtained from the original snapshot.

`IsDBNull` should be called to check for `NULL` values before calling this method.

The returned `OracleBlob` object can be used to safely update the BLOB because the BLOB column has been locked after a call to this method.

Invoking this method internally executes a `SELECT . .FOR UPDATE` statement which locks the row.

Different `WAIT` clauses are appended to the statement, depending on the `wait` value. If the `wait` value is:

- 0
"NOWAIT" is appended at the end of a `SELECT . .FOR UPDATE` statement. The statement executes immediately whether the lock is acquired or not. If the lock is not acquired, an exception is thrown.
- n
"WAIT n" is appended at the end of a `SELECT . .FOR UPDATE` statement. The statement executes as soon as the lock is acquired. However, if the lock cannot be acquired by *n* seconds, this method call throws an exception.

The `WAIT n` feature is only available for Oracle9i or later. For any version lower than Oracle9i, *n* is implicitly treated as -1 and nothing is appended at the end of a `SELECT . .FOR UPDATE` statement.
- -1
Nothing is appended at the end of the `SELECT . .FOR UPDATE`. The statement execution waits indefinitely until a lock can be acquired.

Example

The `GetOracleBlobForUpdate` methods are comparable. See ["Example"](#) for a code example demonstrating usage.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)
- ["LOB Support"](#)

GetOracleBoolean(int)

This method will return an `OracleBoolean` value of the specified column.

Declaration

```
// C#  
public OracleBoolean GetOracleBoolean(int index);
```

Parameter

`index` - The zero-based column index.

Return Value

The `OracleBoolean` value of the column

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type.

Remarks

- This method can be called on the `BOOLEAN` column, any numeric database type column (`NUMBER`, `BINARY_DOUBLE`, `BINARY_FLOAT`, and `FLOAT`), or any scalar character based column (`CHAR`, `NCHAR`, `VARCHAR2`, `NVARCHAR2`)
- For numeric columns, `FALSE` is returned if column value is 0; `TRUE` otherwise.
- For scalar character based column (`CHAR`, `NCHAR`, `VARCHAR2`, `NVARCHAR2`), `GetBoolean` can be called on Valid String literals.

Valid String literals to represent `TRUE` and `FALSE` (case insensitive) are as follows:

- For `TRUE`, valid string literals are (case insensitive) - `true`, `yes`, `on`, `1`, `t`, `y`
- For `FALSE`, valid string literals are (case insensitive) - `false`, `no`, `off`, `0`, `f`, `n`

If the string literals are anything other than those listed above, then an `InvalidCastException` will be returned when trying to read the value.

- `IsDBNull` should be called to check for `NULL` values before calling this method.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetOracleClob

This method returns an OracleClob object of the specified CLOB column.

Declaration

```
// C#  
public OracleClob GetOracleClob(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The OracleClob value of the column.

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

IsDBNull should be called to check for NULL values before calling this method.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)
- ["LOB Support"](#)

GetOracleClobForUpdate

GetOracleClobForUpdate returns an updatable OracleClob object of the specified CLOB column.

Overload List:

- [GetOracleClobForUpdate\(int\)](#)
This method returns an updatable `OracleClob` object of the specified CLOB column.
- [GetOracleClobForUpdate\(int, int\)](#)
This method returns an updatable `OracleClob` object of the specified CLOB column using a `WAIT` clause.

GetOracleClobForUpdate(int)

This method returns an updatable `OracleClob` object of the specified CLOB column.

Declaration

```
// C#  
public OracleClob GetOracleClobForUpdate(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

An updatable `OracleClob`.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

When the `OracleCommand`'s `ExecuteReader()` method is invoked, all the data fetched by the `OracleDataReader` is from a particular snapshot. Therefore, calling an accessor method on the same column always returns the same value. However, the `GetOracleClobForUpdate()` method incurs a database round-trip to obtain a reference to the current CLOB data while also locking the row using the `FOR UPDATE` clause. This means that the `OracleClob` obtained from `GetOracleClob()` can have a different value than the `OracleClob` obtained from `GetOracleClobForUpdate()` since it is not obtained from the original snapshot.

The returned `OracleClob` object can be used to safely update the CLOB because the CLOB column is locked after a call to this method.

Invoking this method internally executes a `SELECT . . FOR UPDATE` statement without a `WAIT` clause. Therefore, the statement can wait indefinitely until a lock is acquired for that row.

`IsDBNull` should be called to check for `NULL` values before calling this method.

Example

The following example gets the OracleClob object for update from the reader, updates the OracleClob object, and then commits the transaction.

```
/* Database Setup, if you have not done so yet.
connect scott/tiger@oracle
CREATE TABLE empInfo (
empno NUMBER(4) PRIMARY KEY,
empName VARCHAR2(20) NOT NULL,
hiredate DATE,
salary NUMBER(7,2),
jobDescription Clob,
byteCodes BLOB
);

Insert into empInfo(EMPNO,EMPNAME,JOBDESCRIPTION,byteCodes) values
(1,'KING','SOFTWARE ENGR', '5657');
Insert into empInfo(EMPNO,EMPNAME,JOBDESCRIPTION,byteCodes) values
(2,'SCOTT','MANAGER', '5960');
commit;

*/
// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class GetOracleClobForUpdateSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        // Get the job description for empno = 1
        string cmdStr = "SELECT JOBDESCRIPTION, EMPNO FROM EMPINFO where EMPNO = 1";
        OracleCommand cmd = new OracleCommand(cmdStr, con);

        // Since we are going to update the OracleClob object, we will
        // have to create a transaction
        OracleTransaction txn = con.BeginTransaction();

        // Get the reader
        OracleDataReader reader = cmd.ExecuteReader();

        // Declare the variables to retrieve the data in EmpInfo
        OracleClob jobDescClob;

        // Read the first row
        reader.Read();

        if (!reader.IsDBNull(0))
        {
            jobDescClob = reader.GetOracleClobForUpdate(0);

            // Close the reader
            reader.Close();
        }
    }
}
```



```
// Update the job description Clob object
char[] jobDesc = "-SALES".ToCharArray();
jobDescClob.Append(jobDesc, 0, jobDesc.Length);

// Now commit the transaction
txn.Commit();
Console.WriteLine("Clob Column successfully updated");
}
else
    reader.Close();

// Close the connection
con.Close();
}
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)
- ["LOB Support"](#)

GetOracleClobForUpdate(int, int)

This method returns an updatable `OracleClob` object of the specified CLOB column using a WAIT clause.

Declaration

```
// C#
public OracleClob GetOracleClobForUpdate(int index, int wait);
```

Parameters

- *index*
The zero-based column index.
- *wait*
The number of seconds the method waits to acquire a lock.

Return Value

An updatable `OracleClob`.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is NULL.

Remarks

When the `OracleCommand`'s `ExecuteReader()` method is invoked, all the data fetched by the `OracleDataReader` is from a particular snapshot. Therefore, calling an accessor method on the same column always returns the same value. However, the `GetOracleClobForUpdate()` method incurs a database round-trip to obtain a reference to the current CLOB data while also locking the row using the `FOR UPDATE` clause. This means that the `OracleClob` obtained from `GetOracleClob()` can have a different value than the `OracleClob` obtained from `GetOracleClobForUpdate()` since it is not obtained from the original snapshot.

Invoking this method internally executes a `SELECT . .FOR UPDATE` statement which locks the row.

The returned `OracleClob` object can be used to safely update the CLOB because the CLOB column is locked after a call to this method.

Different `WAIT` clauses are appended to the statement, depending on the `wait` value. If the `wait` value is:

- 0
"NOWAIT" is appended at the end of a `SELECT . .FOR UPDATE` statement. The statement executes immediately whether the lock is acquired or not. If the lock is not acquired, an exception is thrown.
- *n*
"WAIT *n*" is appended at the end of a `SELECT . .FOR UPDATE` statement. The statement executes as soon as the lock is acquired. However, if the lock cannot be acquired by *n* seconds, this method call throws an exception.

The `WAIT n` feature is only available for Oracle9i or later. For any version lower than Oracle9i, *n* is implicitly treated as -1 and nothing is appended at the end of a `SELECT . .FOR UPDATE` statement.
- -1
Nothing is appended at the end of the `SELECT . .FOR UPDATE`. The statement execution waits indefinitely until a lock can be acquired.

`IsDBNull` should be called to check for NULL values before calling this method.

Example

The `GetOracleClobForUpdate` methods are comparable. See "[Example](#)" for a code example demonstrating usage.

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)
- "[LOB Support](#)"

GetOracleDataReader

This method returns the `OracleDataReader` object corresponding to the `REF CURSOR` type column in the select list.

Declaration

```
// C#  
public OracleDataReader GetOracleDataReader(int index);
```

Parameters

- `index`
The zero-based column index.

Return Value

The `OracleDataReader` object of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

The application can read data from the returned `OracleDataReader` object.

The returned `OracleDataReader` object inherits the `FetchSize`, `InitialLOBFetchSize`, `InitialLONGFetchSize`, and `UseEdmMapping` property values from `OracleDataReader`.

Sample

```
CREATE OR REPLACE FUNCTION refcurfunc1 RETURN SYS_REFCURSOR  
AS  
    VAR_REF SYS_REFCURSOR;  
BEGIN  
    OPEN VAR_REF FOR  
        SELECT *  
        FROM EMP;  
    RETURN VAR_REF;  
END;
```

```
using (OracleConnection con = new OracleConnection("user id = scott; password =  
<PASSWORD>; data source = oracle"))  
{  
    con.Open();  
  
    OracleCommand cmd = con.CreateCommand();  
    cmd.CommandText = "select refcurfunc1() from dual";  
    OracleDataReader reader = cmd.ExecuteReader();
```

```
while (reader.Read())
{
    OracleDataReader refCurReader = reader.GetOracleDataReader();
    while (refCurReader.Read())
    {
        Console.WriteLine("RefCursor Data: " + refCurReader.GetString(0));
    }
    refCurReader.Close();
    refCurReader.Dispose();
}
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetOracleDate

This method returns an OracleDate structure of the specified DATE column.

Declaration

```
// C#
public OracleDate GetOracleDate(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The OracleDate value of the column.

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

IsDBNull should be called to check for NULL values before calling this method.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)
- ["LOB Support"](#)

GetOracleDecimal

This method returns an `OracleDecimal` structure of the specified `NUMBER` column.

Declaration

```
// C#  
public OracleDecimal GetOracleDecimal(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The `OracleDecimal` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetOracleIntervalDS

This method returns an `OracleIntervalDS` structure of the specified `INTERVAL DAY TO SECOND` column.

Declaration

```
// C#  
public OracleIntervalDS GetOracleIntervalDS(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The `OracleIntervalDS` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetOracleIntervalYM

This method returns an `OracleIntervalYM` structure of the specified `INTERVAL YEAR TO MONTH` column.

Declaration

```
// C#  
public OracleIntervalYM GetOracleIntervalYM(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The `OracleIntervalYM` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetOracleRef

This method returns an `OracleRef` object of the specified REF column.

Declaration

```
// C#  
public OracleRef GetOracleRef(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The `OracleRef` object of the specified column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, the `Read` method has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetOracleRefCursor

This method returns the `OracleRefCursor` object corresponding to the `REF CURSOR` type column in the select list.

Declaration

```
// C#  
public OracleRefCursor GetOracleRefCursor(int index);
```

Parameters

- `index`
The zero-based column index.

Return Value

The `OracleRefCursor` object of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

The application can get the `OracleDataReader` object from the returned `OracleRefCursor` object to read data.

The returned `OracleRefCursor` object inherits the `FetchSize`, `InitialLOBFetchSize`, `InitialLONGFetchSize`, and `UseEdmMapping` property values from `OracleDataReader`.

To minimize the number of open server cursors, `OracleRefCursor` objects should be explicitly disposed.

Sample

```
CREATE OR REPLACE FUNCTION refcurfunc1 RETURN SYS_REFCURSOR  
AS  
    VAR_REF SYS_REFCURSOR;  
BEGIN  
    OPEN VAR_REF FOR  
        SELECT *
```



```
        FROM EMP;  
        RETURN VAR_REF;  
    END;
```

```
using (OracleConnection con = new OracleConnection("user id = scott; password =  
<PASSWORD>; data source = oracle"))  
{  
    con.Open();  
  
    OracleCommand cmd = con.CreateCommand();  
    cmd.CommandText = "select refcurfunc1() from dual";  
    OracleDataReader reader = cmd.ExecuteReader();  
  
    while (reader.Read())  
    {  
        OracleRefCursor refCur = reader.GetOracleRefCursor(0);  
        OracleDataReader refCurReader = refCur.GetDataReader();  
        while (refCurReader.Read())  
        {  
            Console.WriteLine("RefCursor Data: " + refCurReader.GetString(0));  
        }  
        refCurReader.Close();  
        refCurReader.Dispose();  
        refCur.Dispose();  
    }  
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetOracleString

This method returns an `OracleString` structure of the specified column.

Declaration

```
// C#  
public OracleString GetOracleString(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The `OracleString` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

If the column is an Oracle REF column, the string returned is a hexadecimal value that represents the REF in the database.

For `VECTOR` columns, the string returned is a JSON string that represents the `VECTOR` in the database. A sparse vector type column will return a JSON string in sparse format.

For ODP.NET managed and core, a JSON formatted string is returned for the cursor result set from Oracle REF `CURSOR` columns.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetOracleTimeStamp

This method returns an `OracleTimeStamp` structure of the Oracle `TimeStamp` column.

Declaration

```
// C#  
public OracleTimeStamp GetOracleTimeStamp(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The `OracleTimeStamp` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

GetOracleTimeStamp is used with the Oracle Type TimeStamp.

IsDBNull should be called to check for NULL values before calling this method.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetOracleTimeStampLTZ

This method returns an OracleTimeStampLTZ structure of the specified Oracle TimeStamp WITH LOCAL TIME ZONE column.

Declaration

```
// C#  
public OracleTimeStampLTZ GetOracleTimeStampLTZ(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The OracleTimeStampLTZ value of the column.

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

GetOracleTimeStampLTZ is used with the Oracle Type TimeStamp with Local Time Zone columns.

IsDBNull should be called to check for NULL values before calling this method.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetOracleTimeStampTZ

This method returns an `OracleTimeStampTZ` structure of the specified Oracle TimeStamp WITH TIME ZONE column.

Declaration

```
// C#  
public OracleTimeStampTZ GetOracleTimeStampTZ(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The `OracleTimeStampTZ` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is NULL.

Remarks

Used with the Oracle Type TimeStamp with Local Time Zone columns

`IsDBNull` should be called to check for NULL values before calling this method.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetOracleXmlType

This method returns an `OracleXmlType` object of the specified XMLType column.

Declaration

```
// C#  
public OracleXmlType GetOracleXmlType(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The OracleXmlType value of the column.

Exceptions

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

IsDBNull should be called to check for NULL values before calling this method.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetOracleValue

This method returns the specified column value as an ODP.NET type.

Declaration

```
// C#  
public object GetOracleValue(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The value of the column as an ODP.NET type.

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

Remarks

If the column is an Oracle object or Oracle collection column and a custom type mapping exists, then a custom type is returned.

If the column is an Oracle REF column, then an `OracleRef` is returned.

For `VECTOR` columns, an `OracleVector` is returned. When retrieving from a sparse `VECTOR` column, you can cast the `OracleVector` to an `OracleSparseVector` object to utilize sparse vectors APIs.

For ODP.NET managed and core, a JSON formatted string is returned for the cursor result set from Oracle REF CURSOR columns.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetOracleValues

This method gets all the column values as ODP.NET types.

Declaration

```
// C#  
public int GetOracleValues(object[] values);
```

Parameters

- *values*
An array of objects to hold the ODP.NET types as the column values.

Return Value

The number of ODP.NET types in the *values* array.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

Remarks

This method provides a way to retrieve all column values rather than retrieving each column value individually.

The number of column values retrieved is the minimum of the length of the *values* array and the number of columns in the result set.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)
- ["LOB Support"](#)

GetOracleVector

This method returns `OracleVector` object of the specified VECTOR column.

Overload List:

- [GetOracleVector\(int\)](#)
This method returns `OracleVector` object of the specified VECTOR column.
- [GetOracleVector\(string\)](#)
This method returns `OracleVector` object of the specified VECTOR column.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetOracleVector(int)

This method returns an `OracleVector` object of the specified VECTOR column.

Declaration

```
// C#  
public OracleVector GetOracleVector(int index);
```

Parameters

index

A zero-based column index.

Return Value

The `OracleVector` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type.

Remarks

If the column value is NULL, then it returns `OracleVector.Null`.

When retrieving from a sparse VECTOR column, you can cast the `OracleVector` to an `OracleSparseVector` object to utilize sparse vectors APIs.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)
- `OracleVector` class
- `OracleSparseVector` class

GetOracleVector(string)

This method returns an `OracleVector` object of the specified VECTOR column.

Declaration

```
// C#  
public OracleVector GetOracleVector (string name);
```

Parameters

name

The specified column name.

Return Value

The `OracleVector` value of the column.

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, `Read()` has not been called or all rows have been read.

IndexOutOfRangeException - Unable to find specified column in result set.

InvalidCastException - The accessor method is invalid for this column type, or the column value is NULL.

Remarks

If the column value is NULL, then it returns `OracleVector.Null`.

When retrieving from a sparse VECTOR column, you can cast the `OracleVector` to an `OracleSparseVector` object to utilize sparse vectors APIs.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)
- OracleVector class
- OracleSparseVector class

GetOrdinal

This method returns the 0-based ordinal (or index) of the specified column name.

Declaration

```
// C#  
public override int GetOrdinal(string name);
```

Parameters

- *name*
The specified column name.

Return Value

The index of the column.

Implements

IDataRecord

Exceptions

InvalidOperationException - The reader is closed.

IndexOutOfRangeException - The column index is invalid.

Remarks

A case-sensitive search is made to locate the specified column by its name. If this fails, then a case-insensitive search is made.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetProviderSpecificFieldType

This method returns the provider-specific type of the specified column.

Declaration

```
// C#public override Type GetProviderSpecificFieldType(int index);
```

Parameters

- *index*
A zero-based column index.

Return Value

The provider-specific type of the specified column. This is a member of the `Oracle.DataAccess.Types` namespace.

Exceptions

`IndexOutOfRangeException` - The column index is invalid.

`InvalidOperationException` - The reader is closed, or the specified column is a UDT but no registered custom type mapping exists for the UDT.

Remarks

`GetProviderSpecificFieldType` returns a type that corresponds to the value the application obtains after invoking the `GetProviderSpecificValue` accessor on the `OracleDataReader`. For example, if the column is a string, this method returns a .NET Type object for an `OracleString`.

If the attribute is a UDT, this method may return any of the following:

- A .NET Type of the custom type, if the column is an Oracle object or Oracle collection column and a custom type mapping exists.
- A .NET Type of `OracleRef` if the column is an Oracle REF.

For VECTOR columns, .NET Type object of an `OracleVector` is returned. Sparse vectors behave the same as dense vectors.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetProviderSpecificValue

This method returns an object that represents the underlying provider-specific value of the specified ordinal.

Declaration

```
// C#  
public override object GetProviderSpecificValue (int index);
```

Parameters

index

A zero-based column index.

Return Value

An Object that is a representation of the underlying provider-specific field type.

Exceptions

IndexOutOfRangeException - The column index is invalid.

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called or all rows have been read.

Remarks

If the column is an Oracle object or collection column, and a custom type mapping exists, a custom type is returned.

If the column is an Oracle REF column, an OracleRef is returned.

For VECTOR columns, an OracleVector is returned. When retrieving from a sparse VECTOR column, you can cast the OracleVector to an OracleSparseVector object to utilize sparse vectors APIs.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetProviderSpecificValues

This method returns an array of objects that represent the underlying provider-specific values.

Declaration

```
// C#  
public override int GetProviderSpecificValues(object [ ] values);
```

Parameters

- *values*
An array of objects.

Return Value

The number of Object instances in the array.

Exceptions

InvalidOperationException - The reader is closed.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetSchemaTable

This method returns a DataTable that describes the column metadata of the OracleDataReader.

Declaration

```
// C#
public override DataTable GetSchemaTable();
```

Return Value

A DataTable that contains the metadata of the result set.

Implements

IDataReader

Exceptions

InvalidOperationException - The connection is closed or the reader is closed.

Remarks

The OracleDataReader.GetSchemaTable method returns the SchemaTable.

OracleDataReader SchemaTable

The OracleDataReader SchemaTable is a DataTable that describes the column metadata of the OracleDataReader.

The value of ColumnSize can show value up to 32K depending on the definition of VARCHAR2, NVARCHAR2, or RAW type columns in the table definition.

The columns of the SchemaTable are in the order shown.

Table 6-87 OracleDataReader SchemaTable

Name	Name Type	Description
ColumnName	System.String	The name of the column.
ColumnOrdinal	System.Int32	The 0-based ordinal of the column.

Table 6-87 (Cont.) OracleDataReader SchemaTable

Name	Name Type	Description
ColumnSize	System.Int64	The maximum possible length of a value in the column. ColumnSize value is determined as follows: - CHAR and VARCHAR2 types: - in bytes - if IsByteSemantic boolean value is true - in characters - if IsByteSemantic boolean value is false - All other types: - in bytes
NumericPrecision	System.Int16	The maximum precision of the column, if the column is a numeric data type. This column has valid values for Oracle NUMBER, Oracle INTERVAL YEAR TO MONTH, and Oracle INTERVAL DAY TO SECOND columns. For all other columns, the value is null.
NumericScale	System.Int16	The scale of the column. This column has valid values for Oracle NUMBER, Oracle INTERVAL DAY TO SECOND, and the Oracle TIMESTAMP columns. For all other columns, the value is null.
IsUnique	System.Boolean	Indicates whether or not the column is unique. true if no two rows in the base table can have the same value in this column, where the base table is the table returned in BaseTableName. IsUnique is guaranteed to be true if one of the following applies in descending order of priority: - the column constitutes a base table primary key by itself and a NOT NULL constraint has been defined on the column - there is a unique constraint or a unique index that applies only to this column and a NOT NULL constraint has been defined on the column - the column is an explicitly selected ROWID IsUnique is false if the column can contain duplicate values in the base table. The default is false. The value of this property is the same for each occurrence of the base table column in the select list.

Table 6-87 (Cont.) OracleDataReader SchemaTable

Name	Name Type	Description
IsKey	System.Boolean	Indicates whether or not the column is a key column. true if the column is one of a set of columns in the rowset that, taken together, uniquely identify the row. The set of columns with IsKey set to true must uniquely identify a row in the rowset. There is no requirement that this set of columns is a minimal set of columns. This set of columns can be generated from one of the following in descending order of priority: • A base table primary key with the following condition: A NOT NULL constraint must be defined on the column or on all of the columns, in the case of a composite primary key. • Any of the unique constraints or unique indexes with the following condition: A NOT NULL constraint must be defined on the column or on all of the columns, in the case of a composite unique constraint or composite unique index. • A base table composite primary key with the following condition: A NULL constraint must be defined on at least one, but not all, of the columns. • Any of the composite unique constraints or composite unique indexes with the following condition: A NULL constraint must be defined on at least one, but not all, of the columns. An explicitly selected ROWID. false if the column is not required to uniquely identify the row. The value of this property is the same for each occurrence of the base table column in the select list.
IsRowID	System.Boolean	true if the column is a ROWID, otherwise false.
BaseColumnName	System.String	The name of the column in the database if an alias is used for the column.
BaseSchemaName	System.String	The name of the schema in the database that contains the column.
BaseTableName	System.String	The name of the table or view in the database that contains the column.
DataType	System.RuntimeType	Maps to the common language runtime type. For DENSE vector columns, set to .NET numeric type array based on the numeric type of the vector data type. If the numeric format of the vector data type is '*', DataType will be typeof(Array). For SPARSE vector columns, set to typeof(string).
ProviderType	Oracle.DataAccess.Client.OracleDbType or Oracle.ManagedDataAccess.Client.OracleDbType	The database column type (OracleDbType) of the column.
AllowDBNull	System.Boolean	true if null values are allowed, otherwise false.
IsAliased	System.Boolean	true if the column is an alias; otherwise false.

Table 6-87 (Cont.) OracleDataReader SchemaTable

Name	Name Type	Description
IsByteSemantic	System.Boolean	IsByteSemantic is: - true if the ColumnSize value uses bytes semantics - false if ColumnSize uses character semantics This value is always true when connected to a database version earlier than Oracle9i.
IsExpression	System.Boolean	true if the column is an expression; otherwise false.
IsHidden	System.Boolean	true if the column is hidden; otherwise false.
IsReadOnly	System.Boolean	true if the column is read-only; otherwise false.
IsLong	System.Boolean	true if the column is a LONG, LONG RAW, BLOB, CLOB, or BFILE; otherwise false.
UdtTypeName	System.String	The type name of the UDT.
IsAutoIncrement	System.Boolean	true if the column assigns values to new rows in fixed increments; otherwise false. <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
IsIdentity	System.Boolean	true if the column is an identity column; otherwise false.
IdentityType	OracleIdentityType	An OracleIdentityType enumeration value that specifies how the identity column values are generated; otherwise DbNull.Value, if the column is not an identity column. <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
DomainSchema	System.String	The database schema name that contains the domain. <i>Requires Oracle AI Database 26ai or higher.</i>
DomainName	System.String	The domain name <i>Requires Oracle AI Database 26ai or higher.</i>
Annotations	System.String	All column annotations returned as a JSON string <i>Requires Oracle AI Database 26ai or higher.</i>
VectorDimensions	System.Int32	The dimensions of the vector data type. Returns 0 if the dimension is '*', that is the Vector is of any dimension.
VectorType	OracleVectorType	The vector data type. The possible values are OracleVectorType.Dense for dense vectors; OracleVectorType.Sparse for sparse vectors; and null for all other data types. <i>Not Available in ODP.NET, Unmanaged Driver</i>
IsFloatingPointNumber	System.Boolean	Specifies whether column values are floating-point numbers or not. true for floating-point number types, such as NUMBER, NUMBER(*), REAL, DOUBLE PRECISION, FLOAT(p), BINARY_DOUBLE, and BINARY_FLOAT. false for fixed-point number types, such as NUMBER(p, [s]), NUMBER(*, s), and DECIMAL, and for any other Oracle data types, such as VARCHAR2 and DATE. <i>Not Available in ODP.NET, Unmanaged Driver</i>

Example

This example creates and uses the SchemaTable from the reader.

```

/* Database Setup, if you have not done so yet.
connect scott/tiger@oracle
CREATE TABLE empInfo (
empno NUMBER(4) PRIMARY KEY,
empName VARCHAR2(20) NOT NULL,
hiredate DATE,
salary NUMBER(7,2),
jobDescription Clob,
byteCodes BLOB
);

Insert into empInfo(EMPNO,EMPNAME,JOBDESCRIPTION,byteCodes) values
(1,'KING','SOFTWARE ENGR', '5657');
Insert into empInfo(EMPNO,EMPNAME,JOBDESCRIPTION,byteCodes) values
(2,'SCOTT','MANAGER', '5960');
commit;

*/
// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class GetSchemaTableSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        string cmdstr = "SELECT EMPNO,EMPNAME FROM EMPINFO where EMPNO = 1";
        OracleCommand cmd = new OracleCommand(cmdstr, con);

        //get the reader
        OracleDataReader reader = cmd.ExecuteReader();

        //get the schema table
        DataTable schemaTable = reader.GetSchemaTable();

        //retrieve the first column info.
        DataRow row = schemaTable.Rows[0];

        //print out the column info
        Console.WriteLine("Column name: " + row["COLUMNNAME"]);
        Console.WriteLine("Precision: " + row["NUMERICPRECISION"]);
        Console.WriteLine("Scale: " + row["NUMERICSCALE"]);
        reader.Close();

        // Close the connection
        con.Close();
    }
}

```


① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetString

This method returns the string value of the specified column.

Declaration

```
// C#  
public override string GetString(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The string value of the column.

Implements

IDataRecord

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

`GetString()` can be called on all character-based column data types. Starting with managed ODP.NET and ODP.NET Core version 19.10, but not available in unmanaged ODP.NET, `GetString()` can be called on all non-binary data types, returning a string representation of the following additional types with the NLS format session setting honored:

- NUMBER
- DATE
- TIMESTAMP
- TIMESTAMP LTZ
- TIMESTAMP TZ
- INTERVAL YEAR TO MONTH

- INTERVAL DAY TO SECOND
- BINARY_DOUBLE
- BINARY_FLOAT
- FLOAT

Call the `IsDBNull` method to check for null values before calling this method.

If the column is an Oracle REF column, the string returned is a hexadecimal string that represents the REF in the database.

For VECTOR columns, the string returned is a JSON string that represents the VECTOR in the database. A sparse vector type column will return a JSON string in sparse format.

For ODP.NET managed and core, a JSON formatted string is returned for the cursor result set from Oracle REF CURSOR columns.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetTimeSpan

This method returns the `TimeSpan` value of the specified INTERVAL DAY TO SECOND column.

Declaration

```
// C#  
public TimeSpan GetTimeSpan(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The `TimeSpan` value of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is NULL.

Remarks

IsDBNull should be called to check for NULL values before calling this method.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetValue

This method returns the column value as a .NET type.

Declaration

```
// C#  
public override object GetValue(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The value of the column as a .NET type.

Implements

IDataRecord

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, all rows have been read, or no valid custom type mapping has been specified for the Oracle Object or Oracle Collection column.

IndexOutOfRangeException - The column index is invalid.

Remarks

If the column is an Oracle Object or an Oracle Collection column, the .NET custom type corresponding to the custom type mapping is returned.

If the column is an Oracle REF column, a hexadecimal value is returned as a .NET string that represents the REF in the database.

If the UDT is NULL, DBNull.Value is returned.

For dense VECTOR columns, this method returns a numeric array or byte array type based on the VECTOR numeric format. BINARY vectors can only return a .NET byte[]. A sparse vector type column will return a .NET string in sparse format. A binary sparse vector returns a JSON-formatted binary sparse vector .NET string.

For ODP.NET managed and core, a JSON formatted string is returned for the cursor result set from Oracle REF CURSOR columns.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetValues

This method gets all the column values as .NET types.

Declaration

```
// C#  
public override int GetValues(object[ ] values);
```

Parameters

- *values*
An array of objects to hold the .NET types as the column values.

Return Value

The number of objects in the *values* array.

Implements

IDataRecord

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

Remarks

This method provides a way to retrieve all column values rather than retrieving each column value individually.

The number of column values retrieved is the minimum of the length of the values array and the number of columns in the result set.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetVectorDimensions

This method returns the number of dimensions of the specified VECTOR column.

Overload List:

- [GetVectorDimensions\(int\)](#)

This method returns the number of dimensions of the specified VECTOR column.

- [GetVectorDimensions\(string\)](#)

This method returns the number of dimensions of the specified VECTOR column.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetVectorDimensions(int)

This method returns the number of dimensions of the specified VECTOR column.

Declaration

```
// C#  
public long[] GetVectorDimensions(int index);
```

Parameters

index

A zero-based column index.

Return Value

The number of dimensions of the VECTOR column.

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

IsDBNull should be called to check for NULL values before calling this method.

This method also works on VECTOR columns with flex (*) dimensions, and returns the number of dimensions of the vector value in the current row.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetVectorDimensions(string)

This method returns the number of dimensions of the specified VECTOR column.

Declaration

```
// C#  
public long[] GetVectorDimensions(string name);
```

Parameters

name

The specified column name.

Return Value

The number of dimensions of the VECTOR column.

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called or all rows have been read.

IndexOutOfRangeException - Unable to find specified column in result set.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

IsDBNull should be called to check for NULL values before calling this method.

This method also works on VECTOR columns with flex (*) dimensions, and returns the number of dimensions of the vector value in the current row.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetVectorNumericFormat(int)

This method returns the OracleDbType value of the VECTOR column numeric format.

Declaration

```
// C#  
public OracleDbType GetVectorNumericFormat(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The numeric format of the VECTOR column.

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

IsDBNull should be called to check for NULL values before calling this method.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetVectorNumericFormat(string)

This method returns the OracleDbType value of the VECTOR column numeric format.

Declaration

```
// C#  
public OracleDbType GetVectorNumericFormat(string name);
```

Parameters

- *name*
The specified column name.

Return Value

The numeric format of the VECTOR column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - Unable to find specified column in result set.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

GetXmlReader

This method returns the contents of an `XMLType` column as an instance of an .NET `XmlTextReader` object.

Declaration

```
// C#  
public XmlReader GetXmlReader(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

A .NET `XmlTextReader`.

Exceptions

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

IsDBNull

This method indicates whether or not the column value is NULL.

Declaration

```
// C#  
public override bool IsDBNull(int index);
```

Parameters

- *index*

The zero-based column index.

Return Value

Returns true if the column is a NULL value; otherwise, returns false.

Implements

IDataRecord

Exceptions

InvalidOperationException - The reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

Remarks

This method should be called to check for NULL values before calling the other accessor methods.

Example

The code example for the OracleDataReader class includes the IsDBNull method. See ["Example"](#).

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

IsDBNullAsync

Overload List:

IsDBNullAsync returns a Task-based asynchronous version of `OracleDataReader.IsDBNull()`.

- [IsDBNullAsync\(int\)](#)

IsDBNullAsyncAsync returns a Task-based asynchronous version of `OracleDataReader.IsDBNullAsync()`.

- [IsDBNullAsync\(int, CancellationToken\)](#)

IsDBNullAsyncAsync returns a Task-based asynchronous version of `OracleDataReader.IsDBNullAsync()`.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

IsDBNullAsync(int)

This method returns a Task-based asynchronous version of `OracleDataReader.IsDBNull(int32)`.

Declaration

```
// C#  
public Task<bool> IsDBNullAsync(int index);
```

Parameter

index - The zero-based column index

Return Value

`Task<bool>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution

Implements

`DbDataReader`

Exceptions

`InvalidOperationException` - The reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

Remarks

This will call into the `NextResultAsync` implementation with argument `cancellationToken` passed as `CancellationToken.None`.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

IsDBNullAsync(int, CancellationToken)

This method returns a Task-based asynchronous version of `OracleDataReader.IsDBNull(int32)`.

Declaration

```
// C#  
public override Task<bool> IsDBNullAsync(int index, CancellationToken cancellationToken);
```

Parameter

- *index* - The zero-based column index
- *cancellationToken* - The input cancellation token which can be used by the application to cancel the task before command timeout occurs

Return Value

`Task<bool>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution

Implements

`DbDataReader`

Exceptions

`InvalidOperationException` - The reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

Example

```
using Oracle.ManagedDataAccess.Client;  
using System;  
using System.Data;  
using System.Threading;  
  
namespace AsyncApp  
{  
    class AsyncDemo  
    {  
        static async Task Main()  
        {  
            string connectionString = "User Id=HR; Password=<PASSWORD>; Data Source=oracle;";  
            OracleConnection oc = new OracleConnection(connectionString);  
            await oc.OpenAsync(CancellationToken.None);  
        }  
    }  
}
```

```
OracleCommand cmd = oc.CreateCommand();
cmd.CommandText = "select * from demo_table";

OracleDataReader reader;

reader = await cmd.ExecuteReaderAsync();
while (await reader.ReadAsync(CancellationToken.None))
{
    if(!(await reader.IsDBNullAsync(0)))
        Console.WriteLine(reader.GetValue(0));
}
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

NextResult

This method advances the data reader to the next result set.

Declaration

```
// C#
public override bool NextResult();
```

Return Value

Returns true if another result set exists; otherwise, returns false.

Implements

IDataReader

Exceptions

InvalidOperationException - The connection is closed or the reader is closed.

Remarks

NextResult is used when reading results from stored procedure execution that return more than one result set.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

NextResultAsync

Overload List:

NextResultAsync returns a Task-based asynchronous version of `OracleDataReader.NextResult()`.

- [NextResultAsync\(\)](#)

NextResultAsync returns a Task-based asynchronous version of `OracleDataReader.NextResult()`.

- [NextResultAsync\(CancellationTokens\)](#)

NextResultAsync returns a Task-based asynchronous version of `OracleDataReader.NextResult()`.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

NextResultAsync()

This method returns a Task-based asynchronous version of `OracleDataReader.NextResult()`.

Declaration

```
// C#  
public Task<bool> NextResultAsync();
```

Return Value

Task<bool> object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution

Implements

DbDataReader

Exceptions

InvalidOperationException - The connection is closed, or the reader is closed.

Remarks

This will call into the NextResultAsync implementation with argument cancellationTokens passed as CancellationTokens.None.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

NextResultAsync(CancellationToken)

This method returns a Task-based asynchronous version of `OracleDataReader.NextResult()`.

Declaration

```
// C#  
public Task<bool> NextResultAsync(CancellationToken cancellationToken);
```

Parameter

cancellationToken - The input cancellation token which can be used by the application to cancel the task before command timeout occurs.

Return Value

Task<bool> object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution

Implements

DbDataReader

Exceptions

InvalidOperationException - The connection is closed, or the reader is closed.

Example

```
using Oracle.ManagedDataAccess.Client;  
using System;  
using System.Data;  
using System.Threading;  
using System.Threading.Tasks;  
  
namespace AsyncApp  
{  
    class AsyncDemo  
    {  
        static async Task Main()  
        {  
            string connectionString = "User Id=HR; Password=<PASSWORD>; Data Source=oracle;";  
            OracleConnection oc = new OracleConnection(connectionString);  
            await oc.OpenAsync(CancellationToken.None);  
  
            OracleCommand cmd = oc.CreateCommand();  
            cmd.CommandText = "BEGIN OPEN :1 FOR SELECT * FROM EMPLOYEES; OPEN :2 FOR SELECT *  
FROM DEPARTMENTS; END; ";  
  
            OracleParameter p1 = cmd.Parameters.Add("1", OracleDbType.RefCursor);
```

```
p1.Direction = ParameterDirection.Output;

OracleParameter p2 = cmd.Parameters.Add("2", OracleDbType.RefCursor);
p2.Direction = ParameterDirection.Output;

OracleDataReader reader = await cmd.ExecuteReaderAsync();

await reader.ReadAsync(CancellationToken.None);
Console.WriteLine(reader.GetValue(0));

Task task = reader.NextResultAsync(CancellationToken.None);
Console.WriteLine("Hello World");
await task;

await reader.ReadAsync(CancellationToken.None);
Console.WriteLine(reader.GetValue(0));
oc.Close();
    }
}
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

Read

This method reads the next row in the result set.

Declaration

```
// C#
public override bool Read();
```

Return Value

Returns true if another row exists; otherwise, returns false.

Implements

IDataReader

Exceptions

InvalidOperationException - The connection is closed or the reader is closed.

Remarks

The initial position of the data reader is before the first row. Therefore, the Read method must be called to fetch the first row. The row that was just read is considered the *current row*. If the OracleDataReader has no more rows to read, it returns false.

Example

The code example for the OracleDataReader class includes the Read method. See ["Example"](#).

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

ReadAsync

Overload List:

ReadAsync returns a Task-based asynchronous version of `OracleDataReader.Read()`.

- [ReadAsync\(\)](#)

ReadAsync returns a Task-based asynchronous version of `OracleDataReader.Read()`.

- [ReadAsync\(CancellationToken\)](#)

ReadAsync returns a Task-based asynchronous version of `OracleDataReader.Read()` with an input `CancellationToken`.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCommand Class](#)
- [OracleCommand Members](#)

ReadAsync()

This method returns a Task-based asynchronous version of `OracleDataReader.Read()`.

Declaration

```
// C#  
public Task<bool> ReadAsync();
```

Return Value

`Task<bool>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution

Implements

`DbDataReader`

Exceptions

`InvalidOperationException` - The connection is closed, or the reader is closed.

Remarks

This will call into the ReadAsync implementation with argument cancellationToken passed as CancellationToken.None.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

ReadAsync(CancellationToken)

This method returns a Task-based asynchronous version of `OracleDataReader.Read()`.

Declaration

```
// C#  
public override Task<bool> ReadAsync(CancellationToken cancellationToken)
```

Parameters

`cancellationToken` - The input cancellation token which can be used by the application to cancel the task before command timeout occurs.

Return Value

`Task<bool>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution

Implements

`DbDataReader`

Exceptions

`InvalidOperationException` - The connection is closed, or the reader is closed.

Example

```
using Oracle.ManagedDataAccess.Client;  
using System;  
using System.Threading;  
  
namespace AsyncApp  
{  
    class AsyncDemo  
    {  
        static async Task Main()  
        {  
            string connectionString = "User Id=HR; Password=<PASSWORD>; Data Source=oracle;";  
            OracleConnection oc = new OracleConnection(connectionString);  
            await oc.OpenAsync(CancellationToken.None);  
  
            OracleCommand cmd = oc.CreateCommand();
```

```
cmd.CommandText = "select * from demo_table";

OracleDataReader reader;
reader = await cmd.ExecuteReaderAsync();
while (await reader.ReadAsync(CancellationToken.None))
{
    Console.WriteLine(reader.GetValue(0));
}
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataReader Class](#)
- [OracleDataReader Members](#)

OracleDataSource Class

An OracleDataSource object represents an Oracle database from which to open connections to or execute commands directly against.

OracleDataSource has no public constructor. To create an instance, use OracleClientFactory.CreateDataSource(string connectionString) or an OracleDataSourceBuilder.

Class Inheritance

System.Object

System.Data.Common.DbDataSource

Oracle.ManagedDataAccess.Client.OracleDataSource

Declaration

```
// C#
public class OracleDataSource : DbDataSource
```

Requirements

Provider	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client
.NET Framework	-
.NET (Core)	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSource Members](#)
- [OracleDataSource Public Methods](#)
- [OracleDataSource Properties](#)
- [OracleDataSource Public Events](#)

OracleDataSource Members

OracleDataSource members are listed in the following tables.

OracleDataSource Public Methods

OracleDataSource methods are listed in [Table 6-88](#).

Table 6-88 OracleDataSource Public Methods

Method	Description
ClearPool	Clears the OracleDataSource associated pool.
CreateCommand	Returns an OracleCommand ready to execute against the OracleDataSource
CreateConnection	Returns a new, closed database connection represented by the OracleDataSource
Dispose	void Dispose() disposes the OracleDataSource and clears the pool associated with it.
DisposeAsync	ValueTask DisposeAsync() asynchronously disposes the OracleDataSource and clears the pool associated with it.
OpenConnection	Returns a new, open database connection represented by the OracleDataSource
OpenConnectionAsync	Asynchronously returns a new, open database connection represented by the OracleDataSource

OracleDataSource Properties

OracleDataSource properties are listed in [Table 6-89](#).

Table 6-89 OracleDataSource Properties

Property	Description
ConnectionString	The connection string associated with the OracleDataSource
DatabaseCharset	Returns the database character set
DatabaseMaxIdentifierLength	Returns the database maximum identifier length
DatabaseNCharset	Returns the database national character set
Password	Updates the OracleDataSource password

Table 6-89 (Cont.) OracleDataSource Properties

Property	Description
ProxyPassword	Updates the OracleDataSource proxy password
RetryCount	Specifies the number of times a TNS ADDRESS list is traversed before the connection attempt times out
RetryDelay	Specifies the delay in seconds between subsequent retries for a connection

OracleDataSource Public Events

OracleDataSource public events are listed in [Table 6-90](#).

Table 6-90 OracleDataSource Public Events

Public Events	Description
ConnectionOpen	OracleConnectionOpenEventHandler ConnectionOpen is triggered upon the OracleDataSource.OpenConnection() method or a connection opening to undertake Application Continuity failover and replay. See also OracleConnection ConnectionOpen event.
InfoMessage	OracleInfoMessageEventHandler InfoMessage is triggered for any message or warning sent by the database. See also OracleConnection InfoMessage event.
HAEvent	OracleHAEventHandler HAEvent is triggered when an HA event occurs. See also OracleConnection HAEvent event.
Failover	OracleFailoverEventHandler Failover is triggered when an Oracle database failover occurs. See also OracleConnection Failover event.
StateChange	StateChangeEventHandler StateChange is triggered when the connection state changes. See also OracleConnection StateChange event.

OracleDataSource Public Methods

OracleDataSource public methods are listed in [Table 6-91](#).

Table 6-91 OracleDataSource Public Methods

Property	Description
ClearPool	Clears the OracleDataSource associated pool.

Table 6-91 (Cont.) OracleDataSource Public Methods

Property	Description
CreateCommand	Returns an <code>OracleCommand</code> ready to execute against the <code>OracleDataSource</code>
CreateConnection	Returns a new, closed database connection represented by the <code>OracleDataSource</code>
Dispose	<code>void Dispose()</code> disposes the <code>OracleDataSource</code> and clears the pool associated with it.
DisposeAsync	<code>ValueTask DisposeAsync()</code> asynchronously disposes the <code>OracleDataSource</code> and clears the pool associated with it.
OpenConnection	Returns a new, open database connection represented by the <code>OracleDataSource</code>
OpenConnectionAsync	Asynchronously returns a new, open database connection represented by the <code>OracleDataSource</code>

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSource Class](#)
- [OracleDataSource Members](#)

ClearPool

This method clears the `OracleDataSource` associated pool.

Declaration

```
// C#  
public void ClearPool()
```

Remarks

This method does nothing if pooling is disabled. `OracleConnection.ClearAllPools` does not clear pools associated with an `OracleDataSource`.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSource Class](#)
- [OracleDataSource Members](#)

CreateCommand

This method returns `OracleCommand` ready to execute against the `OracleDataSource`.

Declaration

```
// C#  
public DbCommand CreateCommand(string command)
```

Parameters

- *command*

The SQL command to execute

Return Value

An `OracleCommand` object ready to execute the command against the `OracleDataSource`.

Remarks

The command returned by this method does not have an explicit `OracleConnection` associated with it. Setting the connection on the `OracleCommand` object returned by this method will throw an exception.

Since the command is not associated with a connection, it cannot participate in transactions. Setting the transaction property on the `OracleCommand` object returned by this method throws an exception.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSource Class](#)
- [OracleDataSource Members](#)

CreateConnection

This method returns a new, closed database connection represented by the `OracleDataSource`.

Declaration

```
// C#  
public DbConnection CreateConnection()
```

Return Value

A closed `OracleConnection` that is represented by the `OracleDataSource`.

Remarks

It is the caller's responsibility to dispose connections obtained through this method.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSource Class](#)
- [OracleDataSource Members](#)

OpenConnection

This method returns a new, open database connection represented by the `OracleDataSource`.

Declaration

```
// C#  
public OracleConnection OpenConnection()
```

Return Value

An open `OracleConnection` that is represented by the `OracleDataSource`.

Remarks

It is the caller's responsibility to dispose connections obtained through this method.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSource Class](#)
- [OracleDataSource Members](#)

OpenConnectionAsync

This method asynchronously returns a new, open database connection represented by the `OracleDataSource`.

Declaration

```
// C#  
public ValueTask<OracleConnection> OpenConnectionAsync(CancellationToken  
    cancellationToken)
```

Parameters

- `cancellationToken`
A token to cancel the asynchronous operation

Return Value

An open `OracleConnection` that is represented by the `OracleDataSource`.

Remarks

It is the caller's responsibility to dispose connections obtained through this method.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDataSource Class](#)
- [OracleDataSource Members](#)

OracleDataSource Properties

OracleDataSource properties are listed in [Table 6-92](#).

Table 6-92 OracleDataSource Properties

Property	Description
ConnectionString	Returns the connection string associated with the OracleDataSource
DatabaseCharset	Returns the database character set
DatabaseMaxIdentifierLength	Returns the database maximum identifier length
DatabaseNCharset	Returns the database national character set
Password	Updates the OracleDataSource password
ProxyPassword	Updates the OracleDataSource proxy password
RetryCount	Specifies the number of times a TNS ADDRESS list is traversed before the connection attempt times out
RetryDelay	Specifies the delay in seconds between subsequent retries for a connection

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDataSource Class](#)
- [OracleDataSource Members](#)

ConnectionString

This property returns the connection string associated with the OracleDataSource.

Declaration

```
// C#  
public string ConnectionString { get; }
```


Remarks

This property returns the connection string without the password when Persist Security Info is set to false.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDataSource Class](#)
- [OracleDataSource Members](#)
- OracleConnection property [ConnectionString](#)

DatabaseCharset

This property returns the database character set.

Declaration

```
// C#  
public OracleDatabaseCharset DatabaseCharset { get; }
```

Remarks

This property can only be accessed after an OracleConnection from this OracleDataSource has been opened. Otherwise, it throws an ORA-50001 error.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDataSource Class](#)
- [OracleDataSource Members](#)

DatabaseMaxIdentifierLength

This property returns the database maximum identifier length.

Declaration

```
// C#  
public int DatabaseMaxIdentifierLength { get; }
```

Remarks

This property can only be accessed after an OracleConnection from this OracleDataSource has been opened. Otherwise, it throws an ORA-50001 error.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSource Class](#)
- [OracleDataSource Members](#)

DatabaseNCharset

This property returns the database national character set.

Declaration

```
// C#  
public OracleDatabaseNCharset DatabaseNCharset { get; }
```

Remarks

This property can only be accessed after an `OracleConnection` from this `OracleDataSource` has been opened. Otherwise, it throws an ORA-50001 error.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSource Class](#)
- [OracleDataSource Members](#)

Password

This property updates the `OracleDataSource` password.

Declaration

```
// C#  
public OracleOpaqueString Password { set; }
```

Exceptions

`ArgumentNullException` - The specified value is null.

Throws "ORA-50029: Password is invalid" if provided password is empty (length of 0).

Throws "ORA-50095: Password should be read-only" if provided `OracleOpaqueString` is not read-only.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSource Class](#)
- [OracleDataSource Members](#)

ProxyPassword

This property updates the OracleDataSource proxy password.

Declaration

```
// C#  
public OracleOpaqueString ProxyPassword { set; }
```

Exceptions

ArgumentNullException - The specified value is null.

Throws "ORA-50029: ProxyPassword is invalid" if provided password is empty (length of 0).

Throws "ORA-50095: ProxyPassword should be read-only" if provided OracleOpaqueString is not read-only.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSource Class](#)
- [OracleDataSource Members](#)

RetryCount

This property specifies the number of times a TNS ADDRESS list is traversed before the connection attempt times out.

Declaration

```
// C#  
public int? RetryCount {get; set;}
```

Exceptions

ArgumentException is thrown when set to a negative value.

Remarks

By default, this property is set to null.

Null setting means other configuration methods determine the client retry count.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSource Class](#)
- [OracleDataSource Members](#)

RetryDelay

This property specifies the delay in seconds between subsequent retries for a connection.

Declaration

```
// C#
public int? RetryDelay {get; set;}
```

Exceptions

ArgumentException is thrown when set to a negative value.

Remarks

By default, this property is set to null.
Null setting means other configuration methods determine the client retry count.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSource Class](#)
- [OracleDataSource Members](#)

OracleDataSource Public Events

OracleDataSource public events are listed in [Table 6-93](#).

Table 6-93 OracleDataSource Public Events

Public Events	Description
ConnectionOpen	OracleConnectionOpenEventHandler ConnectionOpen is triggered upon the OracleDataSource.OpenConnection() method or a connection opening to undertake Application Continuity failover and replay. See also OracleConnection ConnectionOpen event.
InfoMessage	OracleInfoMessageEventHandler InfoMessage is triggered for any message or warning sent by the database. See also OracleConnection InfoMessage event.

Table 6-93 (Cont.) OracleDataSource Public Events

Public Events	Description
HAEvent	OracleHAEventHandler HAEvent is triggered when an HA event occurs. See also OracleConnection HAEvent event.
Failover	OracleFailoverEventHandler Failover is triggered when an Oracle database failover occurs. See also OracleConnection Failover event.
StateChange	StateChangeEventHandler StateChange is triggered when the connection state changes. See also OracleConnection StateChange event.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceCollection Class](#)
- [OracleDataSourceCollection Members](#)

OracleDataSourceBuilder Class

The OracleDataSourceBuilder class creates and configures an OracleDataSource.
During the build, connection configuration is not verified.

Class Inheritance

System.Object

Oracle.ManagedDataAccess.Client.OracleDataSource

Declaration

```
// C#  
public class DbDataSourceBuilder
```

Requirements

Provider	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client
.NET Framework	-
.NET (Core)	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceBuilder Constructor](#)
- [OracleDataSourceBuilder Members](#)
- [OracleDataSourceBuilder Public Methods](#)
- [OracleDataSourceBuilder Properties](#)
- [Methods, Properties, and Events in Common with OracleConnection or OracleDataSource](#)

OracleDataSourceBuilder Members

OracleDataSourceBuilder members are listed in the following tables.

OracleDataSourceBuilder Constructors

OracleDataSourceBuilder constructors are listed in [Table 6-94](#).

Table 6-94 OracleDataSourceBuilder Constructors

Constructor	Description
OracleDataSourceBuilder Constructor	Instantiates a new instance of the OracleDataSourceBuilder class with the provided connection string.

OracleDataSourceBuilder Public Methods

OracleDataSourceBuilder methods are listed in [Table 6-95](#).

Table 6-95 OracleDataSourceBuilder Public Methods

Method	Description
Build	Returns a new OracleDataSource instance.
UsePassword	Sets the OracleDataSource password (Overloaded)
UseProxyPassword	Sets the OracleDataSource proxy user password (Overloaded)

OracleDataSourceBuilder Properties

OracleDataSourceBuilder properties are listed in [Table 6-96](#).

Table 6-96 OracleDataSourceBuilder Properties

Property	Description
ConnectionString	Specifies the OracleDataSource connection string.

OracleDataSourceBuilder Constructor

`OracleDataSourceBuilder(connectionString)` instantiates a new instance of the `OracleDataSourceBuilder` class with the provided connection string.

Declaration

```
public OracleDataSourceBuilder(string connectionString)
```

Parameters

- `connectionString`
The connection information used to connect to the Oracle database.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceBuilder Class](#)
- [OracleDataSourceBuilder Members](#)

OracleDataSourceBuilder Public Methods

`OracleDataSourceBuilder` public methods are listed in [Table 6-97](#).

Table 6-97 OracleDataSourceBuilder Public Methods

Method	Description
Build	Returns a new <code>OracleDataSource</code> instance.
UsePassword	Sets the <code>OracleDataSource</code> password (Overloaded)
UseProxyPassword	Sets the <code>OracleDataSource</code> proxy user password (Overloaded)

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceBuilder Class](#)
- [OracleDataSourceBuilder Members](#)

Build

This method returns a new `OracleDataSource` instance.

Declaration

```
// C#  
public OracleDataSource Build()
```

Return Value

Return Value is a new `OracleDataSource` with the configuration specified by the builder.

Note

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceBuilder Class](#)
- [OracleDataSourceBuilder Members](#)

UsePassword

The `UsePassword` public method sets the `OracleDataSource` password.

Overload List:

- [UsePassword\(password\)](#)
This method sets the `OracleDataSource` password.
- [UsePassword\(opaquePassword\)](#)
This method sets the `OracleDataSource` password.

Note

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceBuilder Class](#)
- [OracleDataSourceBuilder Members](#)

UsePassword(password)

This method sets the `OracleDataSource` password.

Declaration

```
// C#  
public OracleDataSourceBuilder UsePassword(string password)
```


Parameters

- password
The database user password.

Return Value

The OracleDataSourceBuilder itself allowing additional OracleDataSourceBuilder methods to be chained.

Remarks

If a null value is the input parameter, connections will then use the connection string password.

If the password has zero length, ODP.NET throws ArgumentException of ORA-50029.

Note

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceBuilder Class](#)
- [OracleDataSourceBuilder Members](#)

UsePassword(opaquePassword)

This method sets the OracleDataSource password.

Declaration

```
// C#  
public OracleDataSourceBuilder UsePassword(OracleOpaqueString opaquePassword)
```

Parameters

- opaquePassword
The database user password.

Return Value

The OracleDataSourceBuilder itself allowing additional OracleDataSourceBuilder methods to be chained.

Remarks

If a null value is the input parameter, connections will then use the connection string password.

If the password has zero length, ODP.NET throws ArgumentException of ORA-50029.

If opaquePassword is not read only, ODP.NET throws an ORA-50095 error.

Note

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceBuilder Class](#)
- [OracleDataSourceBuilder Members](#)

UseProxyPassword

The UseProxyPassword public method sets the OracleDataSource proxy user password.

Overload List:

- [UseProxyPassword\(proxyPassword\)](#)
This method sets the OracleDataSource proxy user password.
- [UseProxyPassword\(opaquePassword\)](#)
This method sets the OracleDataSource proxy user password.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceBuilder Class](#)
- [OracleDataSourceBuilder Members](#)

UseProxyPassword(proxyPassword)

This method sets the OracleDataSource proxy user password.

Declaration

```
// C#  
public OracleDataSourceBuilder UseProxyPassword(string proxyPassword)
```

Parameters

- proxyPassword
The database proxy user password.

Return Value

The OracleDataSourceBuilder itself allowing additional OracleDataSourceBuilder methods to be chained.

Remarks

If a null value is the input parameter, connections will then use the connection string password.
If the password has zero length, ODP.NET throws ArgumentException of ORA-50029.

 Note

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceBuilder Class](#)
- [OracleDataSourceBuilder Members](#)

UseProxyPassword(opaquePassword)

This method sets the OracleDataSource proxy user password.

Declaration

```
// C#
public OracleDataSourceBuilder UseProxyPassword(OracleOpaqueString opaquePassword)
```

Parameters

- opaquePassword
The database proxy user password.

Return Value

The OracleDataSourceBuilder itself allowing additional OracleDataSourceBuilder methods to be chained.

Remarks

If a null value is the input parameter, connections will then use the connection string password.
If the password has zero length, ODP.NET throws ArgumentException of ORA-50029.
If opaquePassword is not read only, ODP.NET throws an ORA-50095 error.

 Note

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceBuilder Class](#)
- [OracleDataSourceBuilder Members](#)

OracleDataSourceBuilder Properties

OracleDataSourceBuilder properties are listed in [Table 6-98](#).

Table 6-98 OracleDataSourceBuilder Properties

Property	Description
ConnectionString	Specifies the OracleDataSource connection string.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceBuilder Class](#)
- [OracleDataSourceBuilder Members](#)

ConnectionString

This property specifies the OracleDataSource connection string.

Declaration

```
// C#
public string ConnectionString { get; }
```

Property Value

The connection string required to establish a connection to the Oracle database.

① Note

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceBuilder Class](#)
- [OracleDataSourceBuilder Members](#)

Methods, Properties, and Events in Common with OracleConnection or OracleDataSource

OracleDataSourcebuilder allows ODP.NET connections to be configured with many of the same properties and events that are part of OracleConnection or OracleDataSource. The table below shows the OracleDataSourceBuilder setter methods and properties with the counterpart OracleConnection or OracleDataSource properties.

① Note

These OracleDataSourceBuilder methods and properties do not have getters.

OracleDataSourceBuilder Methods	Corresponding OracleConnection or OracleDataSource Property
public OracleDataSourceBuilder UseAccessToken(OracleAccessToken accessToken)	AccessToken

OracleDataSourceBuilder Methods	Corresponding OracleConnection or OracleDataSource Property
public OracleDataSourceBuilder UseActionName(string actionName)	ActionName
public OracleDataSourceBuilder AllowCertificateSelectionUI(bool allowCertificateSelection)	AllowCertificateSelectionUI
public OracleDataSourceBuilder UseAutoCommit(bool useAutoCommit)	AutoCommit
public OracleDataSourceBuilder UseAutoProxy(bool useAutoProxy)	AutoProxy
public OracleDataSourceBuilder UseBindByName(bool useBindByName)	BindByName
public OracleDataSourceBuilder UseChunkMigrationConnectionTimeout(string chunkMigrationConnectionTimeout)	ChunkMigrationConnectionTimeout
public OracleDataSourceBuilder UseClientId(string clientId)	ClientId
public OracleDataSourceBuilder UseClientInfo(string clientInfo)	ClientInfo
public OracleDataSourceBuilder UseCommandTimeout(int commandTimeout)	CommandTimeout
public OracleDataSourceBuilder UseConnectionIdPrefix(string connectionIdPrefix)	ConnectionIdPrefix
public OracleDataSourceBuilder UseDatabaseCharset(OracleDatabaseCharset charset)	DatabaseCharset
public OracleDataSourceBuilder UseDatabaseEditionName(string databaseEditionName)	DatabaseEditionName
public OracleDataSourceBuilder UseDatabaseNCharset(OracleDatabaseNCharset nCharset)	DatabaseNCharset
public OracleDataSourceBuilder UseDRCPConnectionClass(string drcpConnectionClass)	DRCPConnectionClass
public OracleDataSourceBuilder UseDRCPurity(OracleConnection.OracleDRCPurity drcpPurity)	DRCPurity
public OracleDataSourceBuilder UseFetchSize(long fetchSize)	FetchSize
public OracleDataSourceBuilder UseHttpsProxy(string httpsProxy)	HttpsProxy
public OracleDataSourceBuilder UseHttpsProxyPort(int httpsProxyPort)	HttpsProxyPort
public OracleDataSourceBuilder UseKeepAlive(bool useKeepAlive)	KeepAlive

OracleDataSourceBuilder Methods	Corresponding OracleConnection or OracleDataSource Property
<code>public OracleDataSourceBuilder UseKeepAliveInterval(int keepAliveInterval)</code>	KeepAliveInterval
<code>public OracleDataSourceBuilder UseKeepAliveTime(int keepAliveTime)</code>	KeepAliveTime
<code>public OracleDataSourceBuilder UseMaxStatementCacheSize(int maxStatementCacheSize)</code>	MaxStatementCacheSize
<code>public OracleDataSourceBuilder UseModuleName(string moduleName)</code>	ModuleName
<code>public OracleDataSourceBuilder UseNamesDirectoryPath(string namesDirectoryPath)</code>	NamesDirectoryPath
<code>public OracleDataSourceBuilder UseOciCompartment(string ociCompartment)</code>	OciCompartment
<code>public OracleDataSourceBuilder UseOciDatabase(string ociDatabase)</code>	OciDatabase
<code>public OracleDataSourceBuilder UseOciIamUrl(string ociIamUrl)</code>	OciIamUrl
<code>public OracleDataSourceBuilder UseOciTenancy(string ociTenancy)</code>	OciTenancy
<code>public OracleDataSourceBuilder UsePasswordAuthentication(OraclePasswordAuth passwordAuth)</code>	PasswordAuthentication
<code>public OracleDataSourceBuilder UsePdbName(string pdbName)</code>	PDBName
<code>public OracleDataSourceBuilder UsePipelining(bool usePipelining)</code>	Pipelining
<code>public OracleDataSourceBuilder UsePoolName(string poolName)</code>	PoolName
<code>public OracleDataSourceBuilder UseRemoteConfigurationFiltering(bool useRemoteConfigFiltering)</code>	RemoteConfigurationFiltering
<code>public OracleDataSourceBuilder UseRetryCount(int RetryCount)</code>	RetryCount
<code>public OracleDataSourceBuilder UseRetryDelay(int RetryDelay)</code>	RetryDelay
<code>public OracleDataSourceBuilder UseSDU(int SDU)</code>	SDU
<code>public OracleDataSourceBuilder UseServiceName(string serviceName)</code>	ServiceName
<code>public OracleDataSourceBuilder UseServiceRelocationConnectionTimeout(string serviceRelocationTimeout)</code>	ServiceRelocationConnectionTimeout

OracleDataSourceBuilder Methods	Corresponding OracleConnection or OracleDataSource Property
<code>public OracleDataSourceBuilder UseSqlNetAllowedLogonVersionClient(OracleAllowedLogonVersionClient sqlnetAllowedLogonVersionClient)</code>	SqlNetAllowedLogonVersionClient
<code>public OracleDataSourceBuilder UseSqlNetAuthenticationServices(string sqlNetAuthServices)</code>	SqlNetAuthenticationServices
<code>public OracleDataSourceBuilder UseSqlNetCryptoChecksumClient(string sqlNetCryptoChecksumClient)</code>	SqlNetCryptoChecksumClient
<code>public OracleDataSourceBuilder UseSqlNetCryptoTypesChecksumClient(string sqlNetCryptoChecksumTypesClient)</code>	SqlNetCryptoChecksumTypesClient
<code>public OracleDataSourceBuilder UseSqlNetEncryptionClient(string sqlNetEncryptionClient)</code>	SqlNetEncryptionClient
<code>public OracleDataSourceBuilder UseSqlNetEncryptionClientTypes(string sqlNetEncryptionTypesClient)</code>	SqlNetEncryptionTypesClient
<code>public OracleDataSourceBuilder UseSqlnetWalletOverride(bool useSqlnetWalletOverride)</code>	SqlNetWalletOverride
<code>public OracleDataSourceBuilder UseSSLCertificateThumbprint(string sslCertificateThumbprint)</code>	SSLCertificateThumbprint
<code>public OracleDataSourceBuilder UseSSLServerCertDN(string sslServerCertDN)</code>	SSLServerCertDN
<code>public OracleDataSourceBuilder UseSSLServerDNMatch(bool useSSLServerDNMatch)</code>	SSLServerDNMatch
<code>public OracleDataSourceBuilder UseSSLVersion(string SSLVersion)</code>	SSLVersion
<code>public OracleDataSourceBuilder SuppressGetDecimalInvalidCastException(bool suppressGetDecimalInvalidCastException)</code>	SuppressGetDecimalInvalidCastException
<code>public OracleDataSourceBuilder UseTAFMode(OracleTAFMode tafMode)</code>	TAFMode
<code>public OracleDataSourceBuilder UseTnsAdmin(string tnsAdmin)</code>	TnsAdmin
<code>public OracleDataSourceBuilder UseTokenAuthentication(OracleTokenAuth tokenAuth)</code>	TokenAuthentication
<code>public OracleDataSourceBuilder UseTokenLocation(string tokenLocation)</code>	TokenLocation

OracleDataSourceBuilder Methods	Corresponding OracleConnection or OracleDataSource Property
<code>public OracleDataSourceBuilder UseClientInitiatedCQN(bool useCICQN)</code>	UseClientInitiatedCQN
<code>public OracleDataSourceBuilder UseHourOffsetForUnsupportedTimezone(bool useHourOffsetForUnsupportedTimezone)</code>	UseHourOffsetForUnsupportedTimezone
<code>public OracleDataSourceBuilder UseSNI(bool useSNI)</code>	UseSNI
<code>public OracleDataSourceBuilder UseWalletLocation(string walletLocation)</code>	WalletLocation
<code>public OracleDataSourceBuilder UseWalletPassword(OracleOpaqueString walletpassword)</code>	WalletPassword

The table below shows the OracleDataSourceBuilder events with the counterpart OracleConnection events.

OracleDataSourceBuilder Methods	Corresponding OracleConnection Event
<code>public OracleDataSourceBuilder UseConnectionOpenCallback(OracleConnectionOpenEventHandler connectionOpenCallback)</code>	ConnectionOpen
<code>public OracleDataSourceBuilder UseFailoverCallback(OracleFailoverEventHandler oracleFailoverCallback)</code>	Failover
<code>public OracleDataSourceBuilder UseHAEventCallback(OracleHAEventHandler HACallback)</code>	HAEvent
<code>public OracleDataSourceBuilder UseInfoMessageCallback(OracleInfoMessageEventHandler infoMessageCallback)</code>	InfoMessage
<code>public OracleDataSourceBuilder UseStateChangeCallback(StateChangeEvent Handler stateChangeCallback)</code>	StateChange

① See Also

- [OracleDataSourceBuilder Class](#)
- [OracleDataSourceBuilder Members](#)

OracleDataSourceCollection Class

An OracleDataSourceCollection supports adding and deleting network service name (i.e. TNS) entries for ODP.NET's use to connect to an Oracle database.

Class Inheritance

System.Object

Oracle.ManagedDataAccess.Client.OracleDataSourceCollection

Declaration

```
// C#  
public sealed class OracleDataSourceCollection
```

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
using System;  
using Oracle.ManagedDataAccess.Client;  
  
namespace NetCoreApp  
{  
    class DataSourcesExample  
    {  
        static void Main(string[] args)  
        {  
            // Example to configure Data Sources for the ODP.NET Core provider.  
  
            // Add data source through Add method on OracleDataSourceCollection  
            OracleConfiguration.OracleDataSources.Add("orcl1",  
                "(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=host1)(PORT=1234))  
                (CONNECT_DATA=(SERVICE_NAME=oracle)(SERVER=dedicated)))");  
  
            // Add data source through indexer method on OracleDataSourceCollection  
            OracleConfiguration.OracleDataSources["orcl2"] =  
                "(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=host1)(PORT=1234))  
                (CONNECT_DATA=(SERVICE_NAME=oracle)(SERVER=dedicated)))";  
  
            // Get number of data sources configured  
            int numDataSources = OracleConfiguration.OracleDataSources.Count;  
  
            // Get OracleDataSourceCollection object  
            OracleDataSourceCollection dsColl = OracleConfiguration.OracleDataSources;  
  
            // Add server through Add method on OracleDataSourceCollection  
            dsColl.Add("orcl3", "(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=host3)(PORT=1234))  
            (CONNECT_DATA=(SERVICE_NAME=oracle)(SERVER=dedicated)))");  
        }  
    }  
}
```

```

// Add server through indexer method on OracleDataSourceCollection
dsColl["orcl4"] = "(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=host4)(PORT=1234))
(CONNECT_DATA=(SERVICE_NAME=oracle)(SERVER=dedicated)))";

// Remove a data source
OracleConfiguration.OracleDataSources.Remove("db2");

// Get number of data sources configured
numDataSources = OracleConfiguration.OracleDataSources.Count;

// Get value corresponding to a data source.
string dsVal = OracleConfiguration.OracleDataSources["db1"];

OracleConnection orclCon = null;

try
{
    // Open a test connection
    orclCon = new OracleConnection("user id=scott; password=tiger; data
source=orcl3");

    orclCon.Open();
    orclCon.Close();
}
catch (OracleException ex)
{
    Console.WriteLine(ex);
}
finally
{
    // Close the connection
    if (null != orclCon)
        orclCon.Close();
}
}
}
}

```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceCollection Members](#)
- [OracleDataSourceCollection Methods](#)
- [OracleDataSourceCollection Properties](#)

OracleDataSourceCollection Members

OracleDataSourceCollection members are listed in the following tables.

OracleDataSourceCollection Methods

OracleDataSourceCollection methods are listed in [Table 6-99](#).

Table 6-99 OracleDataSourceCollection Methods

Method	Description
Add	Adds unique network service name (TNS) entries in the OracleDataSourceCollection using net service names and complete connect descriptors
Remove	Removes network service name entries from an OracleDataSourceCollection

OracleDataSourceCollection Properties

OracleDataSourceCollection properties are listed in [Table 6-100](#).

Table 6-100 OracleDataSourceCollection Properties

Property	Description
Count	Returns the number of OracleDataSourceCollection elements
This	Returns or sets the network service name entry with the specified network service name

OracleDataSourceCollection Methods

OracleDataSourceCollection methods are listed in [Table 6-101](#).

Table 6-101 OracleDataSourceCollection Methods

Method	Description
Add	Adds unique network service name (TNS) entries in the OracleDataSourceCollection using net service names and complete connect descriptors
Remove	Removes network service name entries from an OracleDataSourceCollection

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceCollection Class](#)
- [OracleDataSourceCollection Members](#)

Add

This method adds unique network service name (TNS) entries in the OracleDataSourceCollection using net service names and complete connect descriptors.

Declaration

```
// C#  
public void Add(string tnsName, string tnsDescriptor)
```

Parameters

- tnsName
 - Type: System.String
 - Network service name used as the ODP.NET connection string Data Source name.
- tnsDescriptor
 - Type: System.String
 - The complete connect descriptor information for this specific net service name.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceCollection Class](#)
- [OracleDataSourceCollection Members](#)

Remove

This method removes network service name entries from an OracleDataSourceCollection.

Declaration

```
// C#  
public void Remove(string tnsName)
```

Parameter

tnsName

- Type: System.String
- Unique network service name to be removed from the collection.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceCollection Class](#)
- [OracleDataSourceCollection Members](#)

OracleDataSourceCollection Properties

OracleDataSourceCollection properties are listed in [Table 6-102](#).

Table 6-102 OracleDataSourceCollection Properties

Property	Description
Count	Returns the number of OracleDataSourceCollection elements
This	Returns or sets the network service name entry with the specified network service name

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceCollection Class](#)
- [OracleDataSourceCollection Members](#)

Count

This is a read-only property that returns the number of OracleDataSourceCollection elements

Declaration

```
// C#
public int Count { get; }
```

Property Value

Type: System.Int32

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceCollection Class](#)
- [OracleDataSourceCollection Members](#)

This

This property returns or sets the network service name entry with the specified network service name.

Declaration

```
// C#
public string this[string tnsName] { get; set; }
```

Parameters

- tnsName
 - Type: System.String

- Unique network service name to be added or updated
- Value
 - Type: `System.String`
 - Connect descriptor to be added or updated.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceCollection Class](#)
- [OracleDataSourceCollection Members](#)

OracleDataSourceCommand Class

OracleDataSourceCommand opens a connection on behalf of an OracleDataSource object, executes a command, closes the connection.

This class has no public constructor. Class instances are obtained by calling `OracleDataSource.CreateCommand()`.

All OracleDataSourceCommand public methods function identically to OracleCommand public methods. The only difference is OracleDataSourceCommand internally opens a connection when an execution method (e.g., `ExecuteNonQuery()`) is called, then closes the connection after the execution completes. Data types that defer data retrieval, such as CLOBs, BLOBs, and REF Cursors, will not work unless the app uses `ExecuteReader()`. This method keeps the connection open until the reader is closed.

The user does not interact with any connections when working with OracleDataSourceCommand.

A few OracleDataSourceCommand public methods have exceptions to this behavior. Those exceptions are noted in each public method's remarks.

Class Inheritance

System.Object

System.Data.Common.DbCommand

Oracle.ManagedDataAccess.Client.OracleCommand

Oracle.ManagedDataAccess.Client.OracleDataSourceCommand

Declaration

```
// C#  
public class OracleDataSourceCommand : OracleCommand
```

Requirements

Provider	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client

Provider	ODP.NET Core
.NET Framework	-
.NET (Core)	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDataSourceCommand Members](#)
- [OracleDataSource Public Methods](#)
- [OracleDataSourceCommand Properties](#)

OracleDataSourceCommand Members

OracleDataSourceCommand members are listed in the following tables.

OracleDataSourceCommand Public Methods

OracleDataSourceCommand methods are listed in [Table 6-103](#).

Table 6-103 OracleDataSourceCommand Public Methods

Method	Description
ExecuteReader	Returns an OracleDataReader. (Overloaded)
ExecuteReaderAsync	Returns an OracleDataReader asynchronously. (Overloaded)

OracleDataSourceCommand Properties

OracleDataSourceCommand properties are listed in [Table 6-104](#).

Table 6-104 OracleDataSourceCommand Properties

Property	Description
Connection	OracleConnection Connection throws a NotSupportedException.
Transaction	OracleTransaction Transaction throws a NotSupportedException.

OracleDataSource Public Methods

OracleDataSourceCommand public methods are listed in [Table 6-105](#).

Table 6-105 OracleDataSourceCommand Public Methods

Method	Description
ExecuteReader	Returns an <code>OracleDataReader</code> . (Overloaded)
ExecuteReaderAsync	Returns an <code>OracleDataReader</code> asynchronously. (Overloaded)

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDataSourceCommand Class](#)
- [OracleDataSourceCommand Members](#)

ExecuteReader

`OracleDataReader` `ExecuteReader` returns an `OracleDataReader`.

Overload List:

- `ExecuteReader()`

Since `OracleDataReader` relies on an open connection to fetch data, ODP.NET does not close the connection after `ExecuteReader()` returns.

ODP.NET sets the `CommandBehavior` to `CommandBehavior.CloseConnection`. As a result, the connection is closed once the reader is closed.

- `ExecuteReader(commandBehavior)`

`commandBehavior` argument is ignored. The `CommandBehavior.CloseConnection` value is used.

Since `OracleDataReader` relies on an open connection to fetch data, ODP.NET does not close the connection after `ExecuteReader()` returns.

ODP.NET sets the `CommandBehavior` to `CommandBehavior.CloseConnection`. As a result, the connection is closed once the reader is closed.

Note

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDataSourceCommand Class](#)
- [OracleDataSourceCommand Members](#)

ExecuteReaderAsync

ExecuteReaderAsync returns an `OracleDataReader` asynchronously.

Overload List:

- [ExecuteReaderAsync\(cancellationToken\)](#)
This method returns an `OracleDataReader` asynchronously.
- [ExecuteReaderAsync\(commandBehavior, cancellationToken\)](#)
This method returns an `OracleDataReader` asynchronously with `CommandBehavior.CloseConnection`.

Note

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceCommand Class](#)
- [OracleDataSourceCommand Members](#)

ExecuteReaderAsync(cancellationToken)

This method returns an `OracleDataReader` asynchronously.

Declaration

```
// C#  
public Task<OracleDataReader> ExecuteReaderAsync(CancellationToken cancellationToken)
```

Parameters

- `cancellationToken`
Token to cancel async call.

Remarks

Since `OracleDataReader` relies on an open connection to fetch data, ODP.NET does not close the connection after `ExecuteReader()` returns.

ODP.NET sets the `CommandBehavior` to `CommandBehavior.CloseConnection`. As a result, the connection is closed once the reader is closed.

Note

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceCommand Class](#)
- [OracleDataSourceCommand Members](#)

ExecuteReaderAsync(commandBehavior, cancellationToken)

This method returns an `OracleDataReader` asynchronously.

Declaration

```
// C#  
public Task<OracleDataReader> ExecuteReaderAsync(CommandBehavior commandBehavior,  
CancellationToken cancellationToken)
```

Parameters

- `commandBehavior`
This argument is ignored. The `CommandBehavior.CloseConnection` value is used.
- `cancellationToken`
Token to cancel async call.

Remarks

Since `OracleDataReader` relies on an open connection to fetch data, ODP.NET does not close the connection after `ExecuteReader()` returns.

ODP.NET sets the `CommandBehavior` to `CommandBehavior.CloseConnection`. As a result, the connection is closed once the reader is closed.

Note

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceCommand Class](#)
- [OracleDataSourceCommand Members](#)

OracleDataSourceCommand Properties

`OracleDataSourceCommand` properties are listed in [Table 6-106](#).

Table 6-106 OracleDataSourceCommand Properties

Property	Description
Connection	<code>OracleConnectionConnection</code> Connection throws a <code>NotSupportedException</code> .
Transaction	<code>OracleTransaction</code> Transaction throws a <code>NotSupportedException</code> .

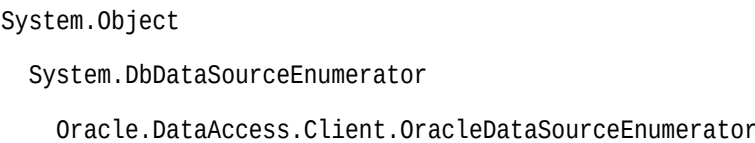
① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceCommand Class](#)
- [OracleDataSourceCommand Members](#)

OracleDataSourceEnumerator Class

An OracleDataSourceEnumerator object allows applications to generically obtain a collection of data sources to connect to.

Class Inheritance



Declaration

```
// C#
public sealed class OracleDataSourceEnumerator : DbDataSourceEnumerator
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#

using System;
using System.Data;
using System.Data.Common;
using Oracle.DataAccess.Client;

class DataSourceEnumSample
{
```

```
static void Main()
{
    string ProviderName = "Oracle.DataAccess.Client";

    DbProviderFactory factory = DbProviderFactories.GetFactory(ProviderName);

    if (factory.CanCreateDataSourceEnumerator)
    {
        DbDataSourceEnumerator dsenum = factory.CreateDataSourceEnumerator();
        DataTable dt = dsenum.GetDataSources();

        // Print the first column/row entry in the DataTable
        Console.WriteLine(dt.Columns[0] + " : " + dt.Rows[0][0]);
        Console.WriteLine(dt.Columns[1] + " : " + dt.Rows[0][1]);
        Console.WriteLine(dt.Columns[2] + " : " + dt.Rows[0][2]);
        Console.WriteLine(dt.Columns[3] + " : " + dt.Rows[0][3]);
        Console.WriteLine(dt.Columns[4] + " : " + dt.Rows[0][4]);
    }
    else
        Console.WriteLine("Data source enumeration is not supported by provider");
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceEnumerator Members](#)
- [OracleDataSourceEnumerator Constructor](#)
- [OracleDataSourceEnumerator Public Methods](#)

OracleDataSourceEnumerator Members

OracleDataSourceEnumerator members are listed in the following tables.

OracleDataSourceEnumerator Constructor

OracleDataSourceEnumerator Public Methods are listed in [Table 6-107](#).

Table 6-107 OracleDataSourceEnumerator Method

Method	Description
OracleDataSourceEnumerator Constructor	Instantiates a new instance of the OracleDataSourceEnumerator class

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceEnumerator Class](#)

OracleDataSourceEnumerator Public Methods

OracleDataSourceEnumerator Public Methods are listed in [Table 6-108](#).

Table 6-108 OracleDataSourceEnumerator Method

Method	Description
GetDataSources	Returns a DataTable object with information on all the TNS alias entries in the tnsnames.ora file and entries retrieved from the LDAP servers configured in ldap.ora if LDAP Naming is enabled

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDataSourceEnumerator Class](#)

OracleDataSourceEnumerator Constructor

OracleDataSourceEnumerator constructor creates new instances of an OracleDataSourceEnumerator class.

Declaration

```
// C#  
public OracleDataSourceEnumerator();
```

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDataSourceEnumerator Class](#)
- [OracleDataSourceEnumerator Members](#)

OracleDataSourceEnumerator Public Methods

The OracleDataSourceEnumerator static method is listed in [Table 6-109](#).

Table 6-109 OracleDataSourceEnumerator Method

Method	Description
GetDataSources	Returns a DataTable object with information on all the TNS alias entries in the tnsnames.ora file

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceEnumerator Class](#)
- [OracleDataSourceEnumerator Members](#)

GetDataSources

This method returns a `DataTable` object with information on all the .NET configuration file ODP.NET data sources, TNS alias entries in the `tnsnames.ora` file, and entries retrieved from the LDAP servers configured in `ldap.ora` if LDAP naming is enabled.

Declaration

```
// C#  
public override DataTable GetDataSources();
```

Return Value

A `DataTable` object.

Remarks

This method returns a `DataTable` object for each ODP.NET data source in the .NET configuration file, TNS alias entry that exists in the `tnsnames.ora` file, and each entry retrieved from the LDAP servers. If there are no ODP.NET data sources in the .NET configuration file, a `tnsnames.ora` file is not found, nor LDAP Naming is configured, then the returned `DataTable` object will be empty.

This method in ODP.NET, Managed Driver can fetch all the data source aliases from an LDAP server, such as Oracle Internet Directory or Microsoft Active Directory. This method in ODP.NET, Unmanaged Driver does not support retrieving data source aliases from an LDAP server.

When Oracle Internet Directory (OID) is used for the TNS naming repository, there is a limit of 1000 TNS entries retrieved.

The following columns are returned for each row, but only the `InstanceName` column is populated.

- `InstanceName` (type: `System.String`)
- `ServerName` (type: `System.String`)
- `ServiceName` (type: `System.String`)
- `Protocol` (type: `System.String`)
- `Port` (type: `System.String`)

If the TNS and/or LDAP information changes for existing pooled connections, then calling `GetDataSources` will not return these changes unless the pools have been cleared.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceEnumerator Class](#)
- [OracleDataSourceEnumerator Members](#)

OracleDBMSAssert Class

An OracleDBMSAssert class is an ODP.NET class whose methods are for validating and sanitizing input values, such as database identifiers and string literals. This class's methods can be used to prevent SQL injection.

Class Inheritance

System.Object

Oracle.ManagedDataAccess.Client.OracleDBMSAssert

Declaration

```
// C#  
public static class OracleDBMSAssert
```

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

Thread Safety

All public static methods are thread-safe.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDBMSAssert Methods](#)

OracleDBMSAssert Methods

OracleDBMSAssert methods are listed in [Table 6-110](#).

Table 6-110 OracleDBMSAssert Methods

Method	Description
EnquoteIdentifier	Validates the identifier and returns the result. (Overloaded)
EnquoteLiteral	Validates the literal and returns the result. (Overloaded)
EnquoteNCharLiteral	Validates the literal, prefixes an N, and returns the result. (Overloaded)

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDBMSAssert Class](#)

EnquoteIdentifier

This method validates the identifier and returns the result.

Overload List:

- [EnquoteIdentifier\(identifier, maxIdentifierLength, charset, bAlwaysQuote, bCapitalize\)](#)
This method validates the identifier, adds double quotes if missing, and returns the result.
- [EnquoteIdentifier\(identifier, bAlwaysQuote, bCapitalize\)](#)
This method adds double quotes to the identifier if missing and returns the result.

Note

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDBMSAssert Class](#)
- [OracleDBMSAssert Methods](#)

EnquoteIdentifier(identifier, maxIdentifierLength, charset, bAlwaysQuote, bCapitalize)

This method validates the identifier, adds double quotes if missing, and returns the result.

Declaration

```
// C#  
public static string EnquoteIdentifier(string identifier, int maxIdentifierLength,  
    OracleDatabaseCharset charset, bool bAlwaysQuote = true, bool bCapitalize =  
    true);
```

Parameters

- `identifier`
The input identifier.

- `maxIdentifierLength`
The maximum allowed identifier length in bytes.
- `charset`
The database character set.
- `bAlwaysQuote`
If true, then the returned identifier is always quoted. If false, then quotes will only be added for a simple identifier that is not already quoted if the case needs to be preserved.
- `bCapitalize`
If true, a simple identifier will be uppercased except when it is already double quoted. If false, it is case-sensitive, and case will be preserved with double quotes, if it contains a lowercase letter.

Return Value

The identifier is returned with double quotes, except for a simple identifier that is not already quoted in either of the following cases:

- `bAlwaysQuote` is false and `bCapitalize` is true.
- `bAlwaysQuote` is false, `bCapitalize` is false, and the identifier is uppercase.

Exceptions

`ArgumentNullException` - Value cannot be null. (Parameter 'identifier')

`ArgumentException` - Value does not fall within the expected range. (Parameter 'maxIdentifierLength')

`OracleException` - Invalid identifier or literal.

Remarks

This method's purpose is to prevent SQL injection.

Simple SQL names are supported as identifiers. In other words, qualified SQL names are not supported. For example, `SCHEMA.TABLE` would be a qualified SQL name.

A simple identifier that is not already quoted will only get quoted if `bAlwaysQuote` is true, or `bCapitalize` is false and it contains a lowercase letter. It will be uppercased if `bCapitalize` is true.

A quoted identifier is returned as is, but the identifier being quoted cannot contain double quotes or the null character.

Other identifiers can contain any characters, except double quotes and the null character. The identifier will be enclosed in double quotes.

Empty identifiers are not allowed.

Leading and trailing white space is not ignored nor trimmed when quoted.

Reserved words will always be quoted.

`maxIdentifierLength` and `charset` are used for identifier validation. An identifier length must be a minimum of one byte and maximum of `maxIdentifierLength` bytes, when represented in the database character set. The double quotes in a quoted identifier are not counted for the identifier length.

ODP.NET uses charset to verify the specified database character set that can represent the identifier. Sometimes alternative representations of quotes can be used to introduce SQL injection. An OracleException will be thrown if the quotes cannot be correctly represented in the database character set.

This table shows EnquoteIdentifier method output for various input values and settings.

Input	Output bCapitalize=true bAlwaysQuote=false	Output bCapitalize=false bAlwaysQuote=false	Notes
TABLE123\$	TABLE123\$	TABLE123\$	No quotes nor uppercasing needed.
table123\$	TABLE123\$	"table123\$"	For bCapitalize=true, uppercasing is needed, but not quotes. For bCapitalize=false, the identifier needs to be treated as lowercase and quoted.
"table123\$"	"table123\$"	"table123\$"	Already quoted and remains quoted.
table123?	"table123?"	"table123?"	? is not allowed in simple identifiers. No uppercasing. Quotes required.
select	"SELECT"	"select"	Reserved words are quoted. For bCapitalize=true, uppercasing is also needed.
table123\$	"table123\$"	"table123\$"	Leading and trailing white space is not ignored nor trimmed. Quotes are required. It will not be uppercased because it is not a simple identifier.

This ODP.NET code sample shows how to use `OracleDBMSAssert.EnquoteIdentifier` to protect the query from SQL injection compromise.

Example

```
// C#
using System;
using System.Data;
using Oracle.ManagedDataAccess.Client;

class EnquoteIdentifierSample
{
    static void Main()
    {
        string constr = "User Id=HR;Password=<PASSWORD>;Data Source=<DATA SOURCE>";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        string tableName "employees";
        string cmdText = "SELECT count(*) FROM " +
OracleDBMSAssert.EnquoteIdentifier(tableName, 128, OracleDatabaseCharset.AL32UTF8,
false);
        // Resulting Command Text: "SELECT count(*) FROM EMPLOYEES"

        OracleCommand cmd = new OracleCommand(cmdText);
        object count = cmd.ExecuteScalar();
        Console.WriteLine("There are " + count + " rows in the table.");

        cmd.Dispose();
        con.Dispose();
    }
}
```

Note

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDBMSAssert Class](#)
- [OracleDBMSAssert Methods](#)

EnquoteIdentifier(identifier, bAlwaysQuote, bCapitalize)

This method adds double quotes to the identifier if missing and returns the result.

Declaration

```
// C#
public static string EnquoteIdentifier(string identifier, bool bAlwaysQuote = true, bool
    bCapitalize = true);
```

Parameters

- `identifier`

The input identifier.

- **bAlwaysQuote**
If true, then the returned identifier is always quoted. If false, then quotes will only be added for a simple identifier that is not already quoted if the case needs to be preserved.
- **bCapitalize**
If true, a simple identifier will be uppercased except when it is already double quoted. If false, it is case-sensitive, and case will be preserved with double quotes, if it contains a lowercase letter.

Return Value

The identifier is returned with double quotes, except for a simple identifier that is not already quoted in either of the following cases:

- **bAlwaysQuote** is false and **bCapitalize** is true.
- **bAlwaysQuote** is false, **bCapitalize** is false, and the identifier is uppercase.

Exceptions

`ArgumentNullException` - Value cannot be null. (Parameter 'identifier')

`OracleException` - Invalid identifier or literal.

Remarks

This method's purpose is to prevent SQL injection.

Simple SQL names are supported as identifiers. In other words, qualified SQL names are not supported. For example, `SCHEMA.TABLE` would be a qualified SQL name.

A simple identifier that is not already quoted will only get quoted if **bAlwaysQuote** is true, or **bCapitalize** is false and it contains a lowercase letter. It will be uppercased if **bCapitalize** is true.

A quoted identifier is returned as is, but the identifier being quoted cannot contain double quotes or the null character.

Other identifiers can contain any characters, except double quotes and the null character. The identifier will be enclosed in double quotes.

Empty identifiers are not allowed.

Leading and trailing white space is not ignored nor trimmed when quoted.

Reserved words will always be quoted.

If validation of the identifier length and representation in the database character set is needed, then use the other `EnquoteIdentifier` overload.

Note

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDBMSAssert Class](#)
- [OracleDBMSAssert Methods](#)

EnquoteLiteral

This method validates the literal and return the result.

Overload List:

- [EnquoteLiteral\(literal, charset\)](#)

This method validates the literal, adds single quotes if required, and return the result.

- [EnquoteLiteral\(literal\)](#)

This method adds single quotes if required, and return the result.

Note

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDBMSAssert Class](#)
- [OracleDBMSAssert Methods](#)

EnquoteLiteral(literal, charset)

This method validates the literal, adds single quotes if required, and return the result.

Declaration

```
// C#  
public static string EnquoteLiteral(string literal, OracleDatabaseCharset charset);
```

Parameters

- `literal`
The input string literal.
- `charset`
The database character set.

Return Value

The literal is returned enclosed with single quotes.

Exceptions

`ArgumentNullException` - Value cannot be null. (Parameter 'identifier')

`OracleException` - Invalid identifier or literal.

Remarks

This method's purpose is to prevent SQL injection.

Any single quote in the literal is replaced with two single quotes for the purposes of escaping, or overloading. For example, the input string, `scott's`, becomes `'scott's'` in the return value.

Empty literals are allowed with ODP.NET adding quotes with this method.

Leading and trailing white space is not ignored nor trimmed when quoted.

ODP.NET uses charset to verify the specified database character set that can represent the literal. Sometimes alternative representations of quotes can be used to introduce SQL injection. An OracleException will be thrown if the quotes cannot be correctly represented in the database character set.

This table shows EnquoteLiteral method output for various input values.

Input	Output	Notes
scott	'scott'	Add surrounding single quotes.
scott's name	'scott''s name'	Add surrounding single quotes. The existing single quote is replaced with two single quotes.
scott	' scott '	Add surrounding single quotes. Leading and trailing white space is not ignored nor trimmed.

Example

This ODP.NET code sample shows how to use OracleDBMSAssert.EnquoteLiteral to protect the query from SQL injection compromise.

```
// C#
using System;
using System.Data;
using Oracle.ManagedDataAccess.Client;

class EnquoteLiteralSample
{
    static void Main()
    {
        string constr = "User Id=HR;Password=<PASSWORD>;Data Source=<DATA SOURCE>";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        string empName "Walsh";
        string cmdText = "SELECT FIRST_NAME FROM EMPLOYEES WHERE LAST_NAME = " +
        OracleDBMSAssert.EnquoteLiteral(empName, OracleDatabaseCharset.AL32UTF8);
        // Resulting Command Text: "SELECT FIRST_NAME FROM EMPLOYEES WHERE LAST_NAME =
        'Walsh'"

        OracleCommand cmd = new OracleCommand(cmdText);
        DataReader reader = cmd.ExecuteReader();
        if(reader.Read())
        {
            Console.WriteLine("The first name for " + empName + " is " + reader.GetString(0));
        }

        reader.Dispose();
        cmd.Dispose();
        con.Dispose();
    }
}
```

```
}
```

Note

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDBMSAssert Class](#)
- [OracleDBMSAssert Methods](#)

EnquoteLiteral(literal)

This method adds single quotes if required, and return the result.

Declaration

```
// C#  
public static string EnquoteLiteral(string literal);
```

Parameters

- `literal`
The input literal.

Return Value

The literal is returned with single quotes.

Exceptions

ArgumentNullException - Value cannot be null. (Parameter 'identifier')

OracleException - Invalid identifier or literal.

Remarks

This method's purpose is to prevent SQL injection.

Any single quote in the literal is replaced with two single quotes for the purposes of escaping, or overloading. For example, the input string, `scott's`, becomes `'scott's'` in the return value.

Empty literals are allowed.

Leading and trailing white space is not ignored nor trimmed when quoted.

If validation using the database character set is required, then use the other `EnquoteLiteral` overload.

Note

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDBMSAssert Class](#)
- [OracleDBMSAssert Methods](#)

EnquoteNCharLiteral

EnquoteNCharLiteral validates the literal, prefixes an N, and returns the result.

Overload List:

- [EnquoteNCharLiteral\(literal, ncharset\)](#)

This method validates the literal, adds single quotes if required, prefixes an N, and returns the result.

- [EnquoteNCharLiteral\(literal\)](#)

This method adds single quotes if required, prefixes an N, and returns the result.

Note

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDBMSAssert Class](#)
- [OracleDBMSAssert Methods](#)

EnquoteNCharLiteral(literal, ncharset)

This method validates the literal, adds single quotes if required, prefixes an N, and returns the result.

Declaration

```
// C#  
public static string EnquoteNCharLiteral(string literal, OracleDatabaseNCharset  
ncharset);
```

Parameters

- `literal`
The input literal.
- `ncharset`
The database national character set.

Return Value

The literal is returned with single quotes prefixed with an N.

Exceptions

ArgumentNullException - Value cannot be null. (Parameter 'identifier')

OracleException - Invalid identifier or literal.

Remarks

This method's purpose is to prevent SQL injection.

Any single quote in the literal is replaced with two single quotes for the purposes of escaping, or overloading. For example, the input string, scott's, becomes N'scott''s' in the return value.

The literal is returned with single quotes prefixed with an N. Any single quote in the literal is replaced with two single quotes. This is called escaping or overloading.

Empty literals are allowed.

Leading and trailing white space is not ignored nor trimmed when quoted.

ODP.NET uses ncharset to verify the specified database national character set that can represent the literal. Sometimes alternative representations of quotes can be used to introduce SQL injection. An OracleException will be thrown if the quotes cannot be correctly represented in the database national character set.

Note

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDBMSAssert Class](#)
- [OracleDBMSAssert Methods](#)

EnquoteNCharLiteral(literal)

This method adds single quotes if required, prefixes an N, and returns the result.

Declaration

```
// C#  
public static string EnquoteNCharLiteral(string literal);
```

Parameters

- literal
The input literal.

Return Value

The literal is returned with single quotes prefixed with an N.

Exceptions

ArgumentNullException - Value cannot be null. (Parameter 'identifier')

OracleException - Invalid identifier or literal.

Remarks

This method's purpose is to prevent SQL injection.

Any single quote in the literal is replaced with two single quotes for the purposes of escaping, or overloading. For example, the input string, scott's, becomes N'scott''s' in the return value.

Empty literals are allowed.

Leading and trailing white space is not ignored nor trimmed when quoted.

If validation using the database character set is required, then use the other EnquoteLiteral overload.

 **Note**

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDBMSAssert Class](#)
- [OracleDBMSAssert Methods](#)

OracleError Class

The OracleError class represents an error reported by Oracle.

Class Inheritance

System.Object

Oracle.DataAccess.Client.OracleError

Declaration

```
// C#
public sealed class OracleError
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	4.8 See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

The `OracleError` class represents a warning or an error reported by Oracle.

If there are multiple errors, ODP.NET only returns the first error message on the stack.

Example

```
// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;

class OracleErrorsSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        // Create an OracleCommand object using the connection object
        OracleCommand cmd = con.CreateCommand();

        try
        {
            cmd.CommandText = "insert into notable values (99, 'MyText')";
            cmd.ExecuteNonQuery();
        }
        catch (OracleException ex)
        {
            Console.WriteLine("Record is not inserted into the database table.");

            foreach (OracleError error in ex.Errors)
            {
                Console.WriteLine("Error Message: " + error.Message);
                Console.WriteLine("Error Source: " + error.Source);
            }
        }
    }
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleError Members](#)
- [OracleError Static Methods](#)
- [OracleError Properties](#)
- [OracleError Methods](#)

OracleError Members

OracleError members are listed in the following tables.

OracleError Static Methods

The OracleError static method is listed in [Table 6-111](#).

Table 6-111 OracleError Static Method

Method	Description
<code>Equals</code>	Inherited from <code>System.Object</code> (Overloaded)

OracleError Properties

OracleError properties are listed in [Table 6-112](#).

Table 6-112 OracleError Properties

Property	Description
ArrayBindIndex	Specifies the row number of errors that occurred during the Array Bind execution
DataSource	Specifies the Oracle service name (TNS name) that identifies the Oracle database
Message	Specifies the message describing the error
Number	Specifies the Oracle error number
Procedure	Specifies the stored procedure that causes the error
Source	Specifies the name of the data provider that generates the error

OracleError Methods

OracleError methods are listed in [Table 6-113](#).

Table 6-113 OracleError Methods

Method	Description
<code>Equals</code>	Inherited from <code>System.Object</code> (Overloaded)
<code>GetHashCode</code>	Inherited from <code>System.Object</code>
<code>GetType</code>	Inherited from <code>System.Object</code>
ToString	Returns a string representation of the OracleError

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleError Class](#)

OracleError Static Methods

The OracleError static method is listed in [Table 6-114](#).

Table 6-114 OracleError Static Method

Method	Description
Equals	Inherited from System.Object (Overloaded)

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleError Class](#)
- [OracleError Members](#)

OracleError Properties

OracleError properties are listed in [Table 6-115](#).

Table 6-115 OracleError Properties

Property	Description
ArrayBindIndex	Specifies the row number of errors that occurred during the Array Bind execution
DataSource	Specifies the Oracle service name (TNS name) that identifies the Oracle database
Message	Specifies the message describing the error
Number	Specifies the Oracle error number
Procedure	Specifies the stored procedure that causes the error
Source	Specifies the name of the data provider that generates the error

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleError Class](#)
- [OracleError Members](#)

ArrayBindIndex

This property specifies the row number of errors that occurred during the Array Bind execution.

Declaration

```
// C#  
public int ArrayBindIndex {get;}
```

Property Value

An int value that specifies the row number for errors that occurred during the Array Bind execution.

Remarks

Default = 0.

This property is used for Array Bind operations only.

ArrayBindIndex represents the zero-based row number at which the error occurred during an Array Bind operation. For example, if an array bind execution causes two errors on the 2nd and 4th operations, two OracleError objects appear in the OracleErrorCollection with the ArrayBindIndex property values 2 and 4 respectively.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleError Class](#)
- [OracleError Members](#)
- ["Array Binding"](#)

DataSource

This property specifies the Oracle service name (TNS name) that identifies the Oracle database.

Declaration

```
// C#  
public string DataSource {get;}
```

Property Value

A string.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleError Class](#)
- [OracleError Members](#)

Message

This property specifies the message describing the error.

Declaration

```
// C#  
public string Message {get;}
```

Property Value

A string.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleError Class](#)
- [OracleError Members](#)

Number

This property specifies the Oracle error number.

Declaration

```
// C#  
public int Number {get;}
```

Property Value

An int.

Remarks

① Note

ODP.NET error numbers have changed in version 23 to better align with standard Oracle error number ranges.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleError Class](#)
- [OracleError Members](#)

Procedure

This property specifies the stored procedure that causes the error.

Declaration

```
// C#  
public string Procedure {get;}
```

Property Value

The stored procedure name.

Remarks

Represents the stored procedure which creates this OracleError object.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleError Class](#)
- [OracleError Members](#)

Source

This property specifies the name of the data provider that generates the error.

Declaration

```
// C#  
public string Source {get;}
```

Property Value

A string.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleError Class](#)
- [OracleError Members](#)

OracleError Methods

OracleError methods are listed in [Table 6-116](#).

Table 6-116 OracleError Methods

Method	Description
Equals	Inherited from System.Object (Overloaded)
GetHashCode	Inherited from System.Object
GetType	Inherited from System.Object
ToString	Returns a string representation of the OracleError

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleError Class](#)
- [OracleError Members](#)

ToString

Overrides Object

This method returns a string representation of the OracleError.

Declaration

```
// C#  
public override string ToString();
```

Return Value

Returns a string with the format Ora- error number: Class.Method name error message stack trace information.

Example

ORA-24333: zero iteration count

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleError Class](#)
- [OracleError Members](#)

OracleErrorCollection Class

An OracleErrorCollection class represents a collection of all errors that are thrown by the Oracle Data Provider for .NET.

Class Inheritance

System.Object
 System.ArrayList
 Oracle.DataAccess.Client.OracleErrorCollection

Declaration

```
// C#
public sealed class OracleErrorCollection : ArrayList
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

A simple ArrayList that holds a list of OracleErrors.

If there are multiple errors, ODP.NET only returns the first error message on the stack.

Example

```
// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;

class OracleErrorCollectionSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        // Create an OracleCommand object using the connection object
        OracleCommand cmd = con.CreateCommand();

        try
        {
```

```

        cmd.CommandText = "insert into notable values (99, 'MyText')";
        cmd.ExecuteNonQuery();
    }
    catch (OracleException ex)
    {
        Console.WriteLine("Record is not inserted into the database table.");

        foreach (OracleError error in ex.Errors)
        {
            Console.WriteLine("Error Message: " + error.Message);
            Console.WriteLine("Error Source: " + error.Source);
        }
    }
}
}

```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleErrorCollection Members](#)
- [OracleErrorCollection Static Methods](#)
- [OracleErrorCollection Properties](#)
- [OracleErrorCollection Public Methods](#)

OracleErrorCollection Members

OracleErrorCollection members are listed in the following tables.

OracleErrorCollection Static Methods

OracleErrorCollection static methods are listed in [Table 6-117](#).

Table 6-117 OracleErrorCollection Static Methods

Method	Description
Equals	Inherited from System.Object (Overloaded)

OracleErrorCollection Properties

OracleErrorCollection properties are listed in [Table 6-118](#).

Table 6-118 OracleErrorCollection Properties

Property	Description
Capacity	Inherited from System.Collections.ArrayList
Count	Inherited from System.Collections.ArrayList
IsReadOnly	Inherited from System.Collections.ArrayList
IsSynchronized	Inherited from System.Collections.ArrayList
Item	Inherited from System.Collections.ArrayList

OracleErrorCollection Public Methods

OracleErrorCollection public methods are listed in [Table 6-119](#).

Table 6-119 OracleErrorCollection Public Methods

Public Method	Description
CopyTo	Inherited from System.Collections.ArrayList
Equals	Inherited from System.Object (Overloaded)
GetHashCode	Inherited from System.Object
GetType	Inherited from System.Object
ToString	Inherited from System.Object

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleErrorCollection Class](#)

OracleErrorCollection Static Methods

The OracleErrorCollection static method is listed in [Table 6-120](#).

Table 6-120 OracleErrorCollection Static Method

Method	Description
Equals	Inherited from System.Object (Overloaded)

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleErrorCollection Class](#)
- [OracleErrorCollection Members](#)

OracleErrorCollection Properties

OracleErrorCollection properties are listed in [Table 6-121](#).

Table 6-121 OracleErrorCollection Properties

Property	Description
Capacity	Inherited from System.Collections.ArrayList

Table 6-121 (Cont.) OracleErrorCollection Properties

Property	Description
Count	Inherited from System.Collections.ArrayList
IsReadOnly	Inherited from System.Collections.ArrayList
IsSynchronized	Inherited from System.Collections.ArrayList
Item	Inherited from System.Collections.ArrayList

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleErrorCollection Class](#)
- [OracleErrorCollection Members](#)

OracleErrorCollection Public Methods

OracleErrorCollection public methods are listed in [Table 6-122](#).

Table 6-122 OracleErrorCollection Public Methods

Public Method	Description
CopyTo	Inherited from System.Collections.ArrayList
Equals	Inherited from System.Object (Overloaded)
GetHashCode	Inherited from System.Object
GetType	Inherited from System.Object
ToString	Inherited from System.Object

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleErrorCollection Class](#)
- [OracleErrorCollection Members](#)

OracleException Class

The OracleException class represents an exception that is thrown when the Oracle Data Provider for .NET encounters an error. Each OracleException object contains at least one OracleError object in the Error property that describes the error or warning.

Class Inheritance

System.Object
 System.Exception
 System.SystemException
 System.Runtime.InteropServices.ExternalException
 System.Data.Common.DbException
 Oracle.DataAccess.Client.OracleException

Declaration

```
// C#  
public sealed class OracleException : SystemException
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

If there are multiple errors, ODP.NET only returns the first error message on the stack.

Example

```
// C#  
  
using System;  
using System.Data;  
using Oracle.DataAccess.Client;  
  
class OracleExceptionSample  
{  
    static void Main()  
    {  
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";  
        OracleConnection con = new OracleConnection(constr);  
        con.Open();  
  
        // Create an OracleCommand object using the connection object
```

```
OracleCommand cmd = con.CreateCommand();

try
{
    cmd.CommandText = "insert into notable values (99, 'MyText')";
    cmd.ExecuteNonQuery();
}
catch (OracleException ex)
{
    Console.WriteLine("Record is not inserted into the database table.");
    Console.WriteLine("Exception Message: " + ex.Message);
    Console.WriteLine("Exception Source: " + ex.Source);
}
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleException Members](#)
- [OracleException Methods](#)
- [OracleException Static Methods](#)
- [OracleException Static Methods](#)
- [OracleException Properties](#)
- [OracleException Methods](#)

OracleException Members

OracleException members are listed in the following tables.

OracleException Static Methods

The OracleException static method is listed in [Table 6-123](#).

Table 6-123 OracleException Static Method

Method	Description
Equals	Inherited from System.Object (Overloaded)

OracleException Properties

OracleException properties are listed in [Table 6-124](#).

Table 6-124 OracleException Properties

Property	Description
DataSource	Specifies the TNS name that contains the information for connecting to an Oracle instance

Table 6-124 (Cont.) OracleException Properties

Property	Description
Errors	Specifies a collection of one or more <code>OracleError</code> objects that contain information about exceptions generated by the Oracle database
<code>HelpLink</code>	Inherited from <code>System.Exception</code>
<code>InnerException</code>	Inherited from <code>System.Exception</code>
IsRecoverable	Specifies whether the current operation producing this exception can succeed if retried
Message	Specifies the error messages that occur in the exception
Number	Specifies the Oracle error number
OracleLogicalTransaction	Returns an <code>OracleLogicalTransaction</code> object for a recoverable error when using Transaction Guard
Procedure	Specifies the stored procedure that cause the exception
Source	Specifies the name of the data provider that generates the error
<code>StackTrace</code>	Inherited from <code>System.Exception</code>
<code>TargetSite</code>	Inherited from <code>System.Exception</code>

OracleException Methods

OracleException methods are listed in [Table 6-125](#).

Table 6-125 OracleException Methods

Method	Description
<code>Equals</code>	Inherited from <code>System.Object</code> (Overloaded)
<code>GetBaseException</code>	Inherited from <code>System.Exception</code>
<code>GetHashCode</code>	Inherited from <code>System.Object</code>
GetObjectData	Sets the serializable info object with information about the exception
<code>GetType</code>	Inherited from <code>System.Object</code>
ToString	Returns the fully qualified name of this exception

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleException Class](#)

OracleException Static Methods

The `OracleException` static method is listed in [Table 6-126](#).

Table 6-126 OracleException Static Method

Method	Description
Equals	Inherited from System.Object (Overloaded)

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleException Class](#)
- [OracleException Members](#)

OracleException Properties

OracleException properties are listed in [Table 6-127](#).

Table 6-127 OracleException Properties

Property	Description
DataSource	Specifies the TNS name that contains the information for connecting to an Oracle instance
Errors	Specifies a collection of one or more OracleError objects that contain information about exceptions generated by the Oracle database
HelpLink	Inherited from System.Exception
InnerException	Inherited from System.Exception
IsRecoverable	Specifies whether the current operation producing this exception can succeed if retried
Message	Specifies the error messages that occur in the exception
Number	Specifies the Oracle error number
OracleLogicalTransaction	Returns an OracleLogicalTransaction object for a recoverable error when using Transaction Guard
Procedure	Specifies the stored procedure that cause the exception
Source	Specifies the name of the data provider that generates the error
StackTrace	Inherited from System.Exception
TargetSite	Inherited from System.Exception

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleException Class](#)
- [OracleException Members](#)

DataSource

This property specifies the TNS name that contains the information for connecting to an Oracle instance.

Declaration

```
// C#  
public string DataSource {get;}
```

Property Value

The TNS name containing the connect information.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleException Class](#)
- [OracleException Members](#)

Errors

This property specifies a collection of one or more `OracleError` objects that contain information about exceptions generated by the Oracle database.

Declaration

```
// C#  
public OracleErrorCollection Errors {get;}
```

Property Value

An `OracleErrorCollection`.

Remarks

The `Errors` property contains at least one instance of `OracleError` objects.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleException Class](#)
- [OracleException Members](#)

IsRecoverable

This property specifies whether the current operation producing this exception can succeed if retried.

Declaration

```
// C#  
public bool IsRecoverable {get;}
```

Property Value

A bool.

Remarks

When a database outage occurs, such as during a network failure, the session becomes unavailable and the client receives an error code. The client can have difficulty determining whether the in-flight operation committed or needs to be resubmitted. Oracle automatically determines whether an in-flight database operation can be recovered or not using the `IsRecoverable` property. If `IsRecoverable` returns true after an outage, then the application can retrieve the current operation status and complete the transaction. If `IsRecoverable` returns false, then the application can rollback the current operation and resubmit the transaction.

This property is often used in conjunction with Transaction Guard.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleException Class](#)
- [OracleException Members](#)
- [Using Transaction Guard to Prevent Logical Corruption](#)

Message

Overrides Exception

This property specifies the error messages that occur in the exception.

Declaration

```
// C#  
public override string Message {get;}
```

Property Value

A string.

Remarks

Message is a concatenation of all errors in the Errors collection. Each error message is concatenated and is followed by a carriage return, except the last one.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleException Class](#)
- [OracleException Members](#)

Number

This property specifies the Oracle error number.

Declaration

```
// C#  
public int Number {get;}
```

Property Value

The error number.

Remarks

This error number can be the topmost level of error generated by Oracle and can be a provider-specific error number.

① Note

ODP.NET error numbers have changed in version 23 to better align with standard Oracle error number ranges. ODP.NET-specific exceptions now use the ORA-500XXX error range.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleException Class](#)
- [OracleException Members](#)

OracleLogicalTransaction

This property will returns an OracleLogicalTransaction object for a recoverable error when using Transaction Guard.

Declaration

```
// C#  
public OracleLogicalTransaction OracleLogicalTransaction {get;}
```

Property Value

An `OracleLogicalTransaction`.

Remarks

`OracleLogicalTransaction` is non-null when both of the following conditions are met:

- Transaction Guard is enabled on the service
- The exception is a recoverable error

`OracleLogicalTransaction` can be used to determine the transaction outcome by looking at the two properties that it exposes: `Committed` and `UserCallCompleted`. If the outcome is not known, then `Committed` and `UserCallCompleted` will be set to null.

If the outcome of a recoverable error could not be determined by ODP.NET and the connection have not participated in a distributed transaction, then the `OracleLogicalTransactionId` property of the `OracleLogicalTransaction` object will be non-null and it can be used to determine the outcome by having the application explicitly call the `OracleLogicalTransaction.GetOutcome` static method, if the database/service is up.

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleException Class](#)
- [OracleException Members](#)
- [OracleLogicalTransaction Class](#)

Procedure

This property specifies the stored procedure that caused the exception.

Declaration

```
// C#  
public string Procedure {get;}
```

Property Value

The stored procedure name.

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleException Class](#)
- [OracleException Members](#)

Source

Overrides Exception

This property specifies the name of the data provider that generates the error.

Declaration

```
// C#
public override string Source {get;}
```

Property Value

The name of the data provider.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleException Class](#)
- [OracleException Members](#)

OracleException Methods

OracleException methods are listed in [Table 6-128](#).

Table 6-128 OracleException Methods

Method	Description
Equals	Inherited from System.Object (Overloaded)
GetBaseException	Inherited from System.Exception
GetHashCode	Inherited from System.Object
GetObjectData	Sets the serializable info object with information about the exception
GetType	Inherited from System.Object
ToString	Returns the fully qualified name of this exception

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleException Class](#)
- [OracleException Members](#)

GetObjectData

Overrides Exception

This method sets the serializable info object with information about the exception.

Declaration

```
// C#  
public override void GetObjectData(SerializationInfo info, StreamingContext  
    context);
```

Parameters

- *info*
A SerializationInfo object.
- *context*
A StreamingContext object.

Remarks

The information includes DataSource, Message, Number, Procedure, Source, and StackTrace.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleException Class](#)
- [OracleException Members](#)

ToString

Overrides Exception

This method returns the fully qualified name of this exception, the error message in the Message property, the InnerException.ToString() message, and the stack trace.

Declaration

```
// C#  
public override string ToString();
```

Return Value

The string representation of the exception.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Client;  
  
class ToStringSample
```

```
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        // Create an OracleCommand object using the connection object
        OracleCommand cmd = con.CreateCommand();

        try
        {
            cmd.CommandText = "insert into notable values (99, 'MyText')";
            cmd.ExecuteNonQuery(); // This will throw an exception
        }
        catch (OracleException ex)
        {
            Console.WriteLine("Record is not inserted into the database table.");
            Console.WriteLine("ex.ToString() : " + ex.ToString());
        }
    }
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleException Class](#)
- [OracleException Members](#)

OracleInfoMessageEventArgs Class

The `OracleInfoMessageEventArgs` class provides event data for the `OracleConnection.InfoMessage` event. When any warning occurs in the database, the `OracleConnection.InfoMessage` event is triggered along with the `OracleInfoMessageEventArgs` object that stores the event data.

Class Inheritance

`System.Object`

`System.EventArgs`

`Oracle.DataAccess.Client.OracleInfoMessageEventArgs`

Declaration

```
// C#
public sealed class OracleInfoMessageEventArgs
```


Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class InfoMessageSample
{
    public static void WarningHandler(object src,
        OracleInfoMessageEventArgs args)
    {
        Console.WriteLine("Source object is: " + src.GetType().Name);
        Console.WriteLine("InfoMessageArgs.Message is " + args.Message);
        Console.WriteLine("InfoMessageArgs.Source is " + args.Source);
    }
    static void Main()
    {
        OracleConnection con = new OracleConnection("User Id=scott;" +
            "Password=tiger;Data Source=oracle;");

        con.Open();

        OracleCommand cmd = con.CreateCommand();

        //Register to the InfoMessageHandler
        cmd.Connection.InfoMessage +=
            new OracleInfoMessageEventHandler(WarningHandler);

        cmd.CommandText =
            "create or replace procedure SelectWithNoInto( " +
            "  empname in VARCHAR2) AS " +
            "BEGIN " +
            "  select * from emp where ename = empname; " +
            "END SelectWithNoInto;";

        // Execute the statement that produces a warning
```

```
cmd.ExecuteNonQuery();

// Clean up
cmd.Dispose();
con.Dispose();
}
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleInfoMessageEventArgs Members](#)
- [OracleInfoMessageEventArgs Static Methods](#)
- [OracleInfoMessageEventArgs Properties](#)
- [OracleInfoMessageEventArgs Public Methods](#)
- ["OracleConnection Class"](#)

OracleInfoMessageEventArgs Members

OracleInfoMessageEventArgs members are listed in the following tables.

OracleInfoMessageEventArgs Static Methods

The OracleInfoMessageEventArgs static methods is listed in [Table 6-129](#).

Table 6-129 OracleInfoMessageEventArgs Static Method

Method	Description
Equals	Inherited from System.Object (Overloaded)

OracleInfoMessageEventArgs Properties

The OracleInfoMessageEventArgs properties are listed in [Table 6-130](#).

Table 6-130 OracleInfoMessageEventArgs Properties

Property	Description
Errors	Specifies the collection of errors generated by the data source
Message	Specifies the error text generated by the data source
Source	Specifies the name of the object that generated the error

OracleInfoMessageEventArgs Public Methods

The OracleInfoMessageEventArgs methods are listed in [Table 6-131](#).

Table 6-131 OracleInfoMessageEventArgs Public Methods

Method	Description
<code>Equals</code>	Inherited from <code>System.Object</code> (Overloaded)
<code>GetHashCode</code>	Inherited from <code>System.Object</code>
<code>GetType</code>	Inherited from <code>System.Object</code>
ToString	Returns the string representation of the current instance

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleInfoMessageEventArgs Class](#)

OracleInfoMessageEventArgs Static Methods

The `OracleInfoMessageEventArgs` static method is listed in [Table 6-132](#).

Table 6-132 OracleInfoMessageEventArgs Static Method

Method	Description
<code>Equals</code>	Inherited from <code>System.Object</code> (Overloaded)

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleInfoMessageEventArgs Class](#)
- [OracleInfoMessageEventArgs Members](#)

OracleInfoMessageEventArgs Properties

The `OracleInfoMessageEventArgs` properties are listed in [Table 6-133](#).

Table 6-133 OracleInfoMessageEventArgs Properties

Property	Description
Errors	Specifies the collection of errors generated by the data source
Message	Specifies the error text generated by the data source
Source	Specifies the name of the object that generated the error

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleInfoMessageEventArgs Class](#)
- [OracleInfoMessageEventArgs Members](#)

Errors

This property specifies the collection of errors generated by the data source.

Declaration

```
// C#  
public OracleErrorCollection Errors {get;}
```

Property Value

The collection of errors.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleInfoMessageEventArgs Class](#)
- [OracleInfoMessageEventArgs Members](#)

Message

This property specifies the error text generated by the data source.

Declaration

```
// C#  
public string Message {get;}
```

Property Value

The error text.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleInfoMessageEventArgs Class](#)
- [OracleInfoMessageEventArgs Members](#)

Source

This property specifies the name of the object that generated the error.

Declaration

```
// C#  
public string Source {get;}
```

Property Value

The object that generated the error.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleInfoMessageEventArgs Class](#)
- [OracleInfoMessageEventArgs Members](#)

OracleInfoMessageEventArgs Public Methods

The OracleInfoMessageEventArgs methods are listed in [Table 6-134](#).

Table 6-134 OracleInfoMessageEventArgs Public Methods

Method	Description
Equals	Inherited from System.Object (Overloaded)
GetHashCode	Inherited from System.Object
GetType	Inherited from System.Object
ToString	Returns the string representation of the current instance

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleInfoMessageEventArgs Class](#)
- [OracleInfoMessageEventArgs Members](#)

ToString

Overrides Object

This method returns the string representation of the current instance.

Declaration

```
// C#
public override string ToString();
```

Return Value

Returns the OracleInfoMessageEventArgs value in a string representation.

Remarks

If the current instance has a null value, the returned string is null.

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleInfoMessageEventArgs Class](#)
- [OracleInfoMessageEventArgs Members](#)

OracleInfoMessageEventHandler Delegate

The OracleInfoMessageEventHandler represents the signature of the method that handles the OracleConnection.InfoMessage event.

Declaration

```
// C#
public delegate void OracleInfoMessageEventHandler(object sender,
    OracleInfoMessageEventArgs eventArgs);
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Parameters

- *sender*
The source of the event.
- *eventArgs*
The OracleInfoMessageEventArgs object that contains the event data.

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- "[InfoMessage](#)"

OracleLogicalTransaction Class

The OracleLogicalTransaction class provides detailed information about the logical transaction status. Applications can conclusively determine the outcome of the running transaction during the last database outage, then act accordingly to commit, complete, or rollback the transaction.

Class Inheritance

System.Object

System.MarshalByRefObject

Oracle.DataAccess.Client.OracleLogicalTransaction

Declaration

```
// C#
public sealed class OracleLogicalTransaction
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleLogicalTransaction Members](#)
- [OracleLogicalTransaction Public Read-Only Properties](#)
- [OracleLogicalTransaction Methods](#)

OracleLogicalTransaction Members

OracleLogicalTransaction members are listed in the following tables.

OracleLogicalTransaction Public Read-Only Properties

OracleLoigcalTransaction public read-only properties are listed in [Table 6-135](#)

Table 6-135 OracleLogicalTransaction Public Read-Only Properties

Property	Description
Committed	Specifies if the transaction was committed or not
ConnectionString	Specifies a subset of the connection string used for the transaction running during the last database outage
LogicalTransactionId	The logical transaction id is used to determine the commit outcome of the last transaction open in a database session following an outage.
UserCallCompleted	Specifies if the transaction completed and that the information returned may be incomplete and/or not all expected work was completed

OracleLogicalTransaction Methods

OracleLoigcalTransaction methods are listed in [Table 6-136](#)

Table 6-136 OracleLogicalTransaction Methods

Property	Description
Dispose	This method releases any resources or memory allocated by the object
GetOutcome	This method retrieves the transaction outcome from the database server. The method will determine whether the transaction committed and completed or not.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleLogicalTransaction Class](#)

OracleLogicalTransaction Public Read-Only Properties

OracleLoigcalTransaction public read-only properties are listed in [Table 6-137](#)

Table 6-137 OracleLogicalTransaction Public Read-Only Properties

Property	Description
Committed	Specifies if the transaction was committed or not
ConnectionString	Specifies a subset of the connection string used for the transaction running during the last database outage
LogicalTransactionId	The logical transaction id is used to determine the commit outcome of the last transaction open in a database session following an outage.
UserCallCompleted	Specifies if the transaction completed and that the information returned may be incomplete and/or not all expected work was completed

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleLogicalTransaction Class](#)
- [OracleLogicalTransaction Members](#)

Committed

This property specifies if the transaction was committed or not.

Declaration

```
// C#  
public bool? Committed {get;}
```

Property Value

bool.

Remarks

If `GetOutcome()` is not called, the this property holds a null value.

Once `GetOutcome()` is called, then this property will hold either true or false.

In some cases, `OracleLogicalTransaction.GetOutcome` will be called implicitly and populate this property with a non-null value automatically.

[Table 6-138](#) describes the possible outcomes of the `Committed` and `UserCallCompleted` properties.

Table 6-138 Outcome of OracleLogicalTransaction Committed and UserCallCompleted Properties

Committed Value	UserCallCompleted Value	Outcome
false	false	The call did not execute the commit.
true	true	The call did execute the commit and there was no additional information to return and no more work to do if that call was a PL/SQL procedure.
true	false	The transaction is committed, but the information returned may be incomplete and/or not all expected work was completed. Examples of incomplete information or incomplete work done include: the number of rows modified when using autocommit or commit on success, parameter and function results when calling PL/SQL procedures, or PL/SQL procedures with more work to do after the commit. In order to function correctly, .NET applications that use data returned from the commit must check the UserCallCompleted value.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleLogicalTransaction Class](#)
- [OracleLogicalTransaction Members](#)

ConnectionString

This property specifies a subset of the connection string used for the transaction running during the last database outage.

Declaration

```
// C#
public string ConnectionString {get;}
```

Property Value

The data source as a string.

Remarks

This connection string can be useful if the outcome is not known at the time the exception is thrown due to a service that is down. In such a scenario, use the connection string from this property along with the LogicalTransactionId to determine the outcome of the logical transaction by invoking the static GetOutcome() method, once the database or service is back up.

The string returned by this property will contain only the following attributes: User Id, Proxy user Id (if not null/empty), Data Source, and Pooling (which will be set to false).

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleLogicalTransaction Class](#)
- [OracleLogicalTransaction Members](#)

LogicalTransactionId

The logical transaction id is used to determine the commit outcome of the last transaction open in a database session following an outage.

Declaration

```
// C#  
public byte LogicalTransactionId {get;}
```

Property Value

byte[]

Remarks

This logical transaction id can be useful if the outcome is not known at the time the exception is thrown due to a service that is down. In such a scenario, use the byte[] returned from this property (along with the ConnectionString) to determine the outcome of the logical transaction by invoking the static GetOutcome() method, once the database or service is back up.

This property will return a non-null value *only* when the outcome is not known. For example when database or service is down, then the outcome is not known.

LogicalTransactionId property will return null if the connection has participated in a distributed transaction.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleLogicalTransaction Class](#)
- [OracleLogicalTransaction Members](#)

UserCallCompleted

This property specifies if the transaction completed and that the information returned may be incomplete and/or not all expected work was completed.

Declaration

```
// C#  
public bool? UserCallCompleted {get;}
```

Property Value

bool

Remarks

If `GetOutcome()` is not called, the this property holds a null value.

Once `GetOutcome()` is called, then this property will hold either true or false.

In some cases, [GetOutcome](#) will be called implicitly and populate this property with a non-null value automatically

[Table 6-138](#) describes the possible outcomes of the `Committed` and `UserCallCompleted` properties.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleLogicalTransaction Class](#)
- [OracleLogicalTransaction Members](#)

OracleLogicalTransaction Methods

OracleLoigcalTransaction methods are listed in [Table 6-139](#)

Table 6-139 OracleLogicalTransaction Methods

Property	Description
Dispose	This method releases any resources or memory allocated by the object
GetOutcome	This method retrieves the transaction outcome from the database server. The method will determine whether the transaction committed and completed or not.

Dispose

This method releases any resources or memory allocated by the object

Declaration

```
// C#
public void Dispose();
```

Implements

IDisposable

Remarks

The `Dispose` method also closes the `OracleLogicalTransaction` object.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleLogicalTransaction Class](#)
- [OracleLogicalTransaction Members](#)

GetOutcome

The `GetOutcome` method retrieves the transaction outcome from the database server. The method will determine whether the transaction committed and completed or not.

Overload List:

- `GetOutcome(string constring, byte[] ltxid, out bool? bCommitted, out bool? bUserCallCompleted)`

The application can use this static method to determine the outcome if the outcome was not known when the exception was raised.

The application will need to obtain the connection string and logical transaction id from the `OracleException.OracleLogicalTransaction` object before calling this method.

The supplied connection string will be used to establish a connection to the database to determine the outcome of the provided logical transaction id.

ODP.NET implicitly calls `GetOutcome` under the following conditions:

- Transaction Guard is enabled on the service
- `OracleException` is raised
- The exception is a recoverable error

When all of the above is true, then the `OracleException.OracleLogicalTransaction` property will be non-null.

If a connection is involved in a distributed transaction, then `GetOutcome` is *not* called implicitly and the `OracleException.OracleLogicalTransaction.LogicalTransactionId` property returns null.

① Note

Once one server round-trip is incurred for the `GetOutcome()` invocation, the PL/SQL `ForceOutcome` is never invoked again against the server for a given `OracleLogicalTransaction` object.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleLogicalTransaction Class](#)
- [OracleLogicalTransaction Members](#)

OracleOnsServerCollection Class

The OracleOnsServerCollection class supports adding to and deleting from a collection of logical servers with their corresponding list of nodes where the Oracle Notification Service (ONS) daemons are talking to their remote clients.

In case of remote configuration, the application has to specify the <host>:<port> values for every potential database that it can connect to. The <host>:<port> value pairs represent the ports on the different nodes.

Class Inheritance

System.Object

Oracle.ManagedDataAccess.Client.OracleOnsServerCollection

Declaration

```
// C#  
public static class OracleOnsServerCollection
```

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
using System;  
using Oracle.ManagedDataAccess.Client;  
  
namespace NetCoreApplication  
{  
    class OnsServersExample  
    {  
        static void Main(string[] args)  
        {  
            // Example to configure ONS Servers for the ODP.NET Core provider.  
  
            // Add server through Add method on OracleOnsServerCollection  
            OracleConfiguration.OracleOnsServers.Add("db1", "nodeList=host1:port1,  
host2:port2, host3:port3");  
  
            // Add server through indexer method on OracleOnsServerCollection  
            OracleConfiguration.OracleOnsServers["db2"] = "nodeList=m1:p1, m2:p2";  
        }  
    }  
}
```

```

// Get number of servers configured
int numServers = OracleConfiguration.OracleOnsServers.Count;

// Get OracleOnsServerCollection object
OracleOnsServerCollection serverColl = OracleConfiguration.OracleOnsServers;

// Add server through Add method on OracleOnsServerCollection
serverColl.Add("db3", "nodeList=host1:port1, host2:port2, host3:port3");

// Add server through indexer method on OracleOnsServerCollection
serverColl["db4"] = "nodeList=m1:p1, m2:p2";

// Remove a server
OracleConfiguration.OracleOnsServers.Remove("db2");

// Get number of servers configured
numServers = OracleConfiguration.OracleOnsServers.Count;

// Get value corresponding to a server.
string serverVal = OracleConfiguration.OracleOnsServers["db1"];

OracleConnection orclCon = null;

try
{
    // Open a test connection
    orclCon = new OracleConnection("user id=scott; password=tiger; data
source=oracle");

    orclCon.Open();
    orclCon.Close();
}
catch (OracleException ex)
{
    Console.WriteLine(ex);
}
finally
{
    // Close the connection
    if (null != orclCon)
        orclCon.Close();
}
}
}
}

```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleOnsServerCollection Members](#)
- [OracleOnsServerCollection Methods](#)
- [OracleOnsServerCollection Properties](#)

OracleOnsServerCollection Members

OracleOnsServerCollection members are listed in the following tables.

OracleOnsServerCollection Methods

OracleOnsServerCollection methods are listed in [Table 6-140](#).

Table 6-140 OracleOnsServerCollection Methods

Method	Description
Add	Adds unique entries in the OracleOnsServerCollection using logical server names and it's corresponding list of nodes
Remove	Removes an entry from an OracleOnsServerCollection

OracleOnsServerCollection Properties

OracleOnsServerCollection properties are listed in [Table 6-141](#).

Table 6-141 OracleOnsServerCollection Properties

Property	Description
Count	Returns the number of OracleOnsServerCollection elements
This	Returns or sets the an entry for a unique logical server name, which is to be associated with a list of nodes

OracleOnsServerCollection Methods

OracleOnsServerCollection methods are listed in [Table 6-142](#).

Table 6-142 OracleOnsServerCollection Methods

Method	Description
Add	Adds unique entries in the OracleOnsServerCollection using logical server names and it's corresponding list of nodes
Remove	Removes an entry from an OracleOnsServerCollection

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleOnsServerCollection Class](#)
- [OracleOnsServerCollection Members](#)

Add

This method adds unique entries in the OracleOnsServerCollection using logical server names and it's corresponding list of nodes.

Declaration

```
// C#  
public void Add(string serverName, string nodeList)
```

Parameters

- `serverName`
 - Type: `System.String`
 - A unique logical name for a list of nodes to be added
- `nodeList`
 - Type: `System.String`
 - Node names and remote ports for the list of nodes that ODP.NET shares ONS communication with using the following example format: `nodeList=racnode1:4200, racnode2:4200`

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleOnsServerCollection Class](#)
- [OracleOnsServerCollection Members](#)

Remove

This method removes an entry from an `OracleOnsServerCollection`.

Declaration

```
// C#  
public void Remove(string serverName)
```

Parameters

`serverName`

- Type: `System.String`
- A unique logical name for a list of nodes to be removed

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleOnsServerCollection Class](#)
- [OracleOnsServerCollection Members](#)

OracleOnsServerCollection Properties

OracleOnsServerCollection properties are listed in [Table 6-143](#).

Table 6-143 OracleOnsServerCollection Properties

Property	Description
Count	Returns the number of OracleOnsServerCollection elements
This	Returns or sets the an entry for a unique logical server name, which is to be associated with a list of nodes

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataSourceCollection Class](#)
- [OracleDataSourceCollection Members](#)

Count

This is a read-only property that returns the number of OracleOnsServerCollection elements.

Declaration

```
// C#
public int Count { get; }
```

Property Value

Type: System.Int32

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleOnsServerCollection Class](#)
- [OracleOnsServerCollection Members](#)

This

This property returns or sets the an entry for a unique logical server name, which is to be associated with a list of nodes.

Declaration

```
// C#
public string this[string serverName] { get; set; }
```

Parameters

- `serverName`
 - Type: `System.String`
 - Unique logical server name to be added or updated
- `Value`
 - Type: `System.String`
 - Node names and remote ports for the list of nodes that ODP.NET shares ONS communication with using the following example format: `nodeList=racnode1:4200, racnode2:4200`

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleOnsServerCollection Class](#)
- [OracleOnsServerCollection Members](#)

OracleOpaqueString Class

This class securely obfuscates a series of characters, such as passwords. It is like .NET `SecureString` without `SecureString`'s known security weaknesses. Oracle recommends using `OracleOpaqueString` over .NET `SecureString`.

Class Inheritance

`System.Object`

`Oracle.ManagedDataAccess.Client.OracleOpaqueString`

Declaration

```
// C#  
public class OracleOpaqueString
```

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	<code>Oracle.ManagedDataAccess.dll</code>	<code>Oracle.ManagedDataAccess.dll</code>
Namespace	<code>Oracle.ManagedDataAccess.Client</code>	<code>Oracle.ManagedDataAccess.Client</code>
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleOpaqueString Members](#)
- [OracleOpaqueString Constructor](#)
- [OracleOpaqueString Properties](#)
- [OracleOpaqueString Methods](#)

OracleOpaqueString Members

OracleOpaqueString members are listed in the following tables.

OracleOpaqueString Constructors

The OracleOpaqueString constructors are listed in [Table 6-144](#).

Table 6-144 OracleOpaqueString Constructors

Constructors	Description
OracleOpaqueString Constructor	Creates a new OracleOpaqueString instance to securely store text

OracleOpaqueString Methods

The OracleOpaqueString methods are listed in [Table 6-145](#).

Table 6-145 OracleOpaqueString Methods

Method	Description
AppendChar	Obfuscates and appends a new character to the end of the OracleOpaqueString
Clear	Clears all the OracleOpaqueString characters
Copy	Creates a copy of the current OracleOpaqueString
Dispose	Scrubs and releases all OracleOpaqueString resources
Equals(Object)	Inherited from System.Object(Overloaded)
IsReadOnly	Indicates whether the OracleOpaqueString is read-only
MakeReadOnly	Makes the OracleOpaqueString no longer modifiable

OracleOpaqueString Properties

The OracleOpaqueString properties are listed in [Table 6-146](#).

Table 6-146 OracleOpaqueString Properties

Property	Description
Length	Returns the character length of the OracleOpaqueString instance

OracleOpaqueString Constructor

The OracleOpaqueString constructor creates a new OracleOpaqueString instance to securely store text.

Declaration

```
// C#  
public OracleOpaqueString();
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleOpaqueString Class](#)
- [OracleOpaqueString Members](#)

OracleOpaqueString Properties

OracleOpaqueString properties are listed in [Table 6-147](#).

Table 6-147 OracleOpaqueString Properties

Property	Description
Length	Returns the character length of the OracleOpaqueString instance

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleOpaqueString Class](#)
- [OracleOpaqueString Members](#)

Length

This property returns the character length of the OracleOpaqueString instance.

Declaration

```
// C#  
public int Length {get;}
```

Property Value

The OracleOpaqueString character length

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleOpaqueString Class](#)
- [OracleOpaqueString Members](#)

OracleOpaqueString Methods

OracleOpaqueString methods are listed in [Table 6-148](#).

Table 6-148 OracleOpaqueString Methods

Method	Description
AppendChar	Obfuscates and appends a new character to the end of the OracleOpaqueString
Clear	Clears all the OracleOpaqueString characters
Copy	Creates a copy of the current OracleOpaqueString
Dispose	Scrubs and releases all OracleOpaqueString resources
<code>Equals(Object)</code>	Inherited from System.Object(Overloaded)
IsReadOnly	Indicates whether the OracleOpaqueString is read-only
MakeReadOnly	Makes the OracleOpaqueString no longer modifiable

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleOpaqueString Class](#)
- [OracleOpaqueString Members](#)

AppendChar

This method obfuscates and appends a new character to the end of the OracleOpaqueString.

Declaration

```
// C#  
public void AppendChar(Char char)
```

Parameters

- char
The character to append

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleOpaqueString Class](#)
- [OracleOpaqueString Members](#)

Clear

This method clears all the OracleOpaqueString characters.

Declaration

```
// C#  
public void Clear()
```

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleOpaqueString Class](#)
- [OracleOpaqueString Members](#)

Copy

This method creates a copy of the current OracleOpaqueString.

Declaration

```
// C#  
public OracleOpaqueString Copy()
```

Return Value

OracleOpaqueString

Remarks

This method does not have its char[] value in clear text at any point to preserve security.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleOpaqueString Class](#)
- [OracleOpaqueString Members](#)

Dispose

This method scrubs and releases all OracleOpaqueString resources.

Declaration

```
// C#  
public void Dispose()
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleOpaqueString Class](#)
- [OracleOpaqueString Members](#)

IsReadOnly

This method indicates whether the OracleOpaqueString is read-only.

Declaration

```
// C#  
public bool IsReadOnly()
```

Remarks

true if this string is marked read-only; otherwise, false.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleOpaqueString Class](#)
- [OracleOpaqueString Members](#)

MakeReadOnly

This method makes the OracleOpaqueString no longer modifiable.

Declaration

```
// C#
public void MakeReadOnly()
```

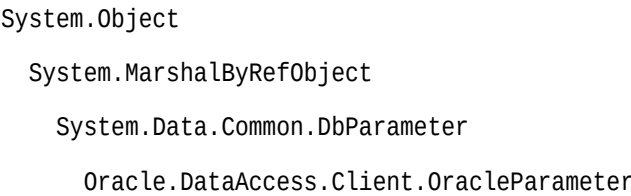
① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleOpaqueString Class](#)
- [OracleOpaqueString Members](#)

OracleParameter Class

An OracleParameter object represents a parameter for an OracleCommand or a DataSet column.

Class Inheritance



Declaration

```
// C#
public sealed class OracleParameter : DbParameter, IDisposable, ICloneable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Exceptions

ArgumentException - The type binding is invalid.

Binding Vector Parameters

The following input Vector parameter data types can be bound as various OracleDbType and OracleDbTypeEx types:

Input Type	OracleDbType Binding Type
.NET numeric array	Vector
	Vector_Int8
	Vector_Float32
	Vector_Float64
.NET byte array	Vector
	Vector_Binary
	Vector_Int8
	Vector_Float32
	Vector_Float64
.NET string	Vector
	Vector_Int8
	Vector_Float32
	Vector_Float64
	Clob
	Varchar2
OracleString	Vector
	Vector_Int8
	Vector_Float32
	Vector_Float64
	Clob
	Varchar2

Example

```
// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class OracleParameterSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleParameter[] prm = new OracleParameter[3];

        // Create OracleParameter objects through OracleParameterCollection
        OracleCommand cmd = con.CreateCommand();

        cmd.CommandText = "select max(empno) from emp";
        int maxno = int.Parse(cmd.ExecuteScalar().ToString());
    }
}
```

```
prm[0] = cmd.Parameters.Add("paramEmpno", OracleDbType.Decimal,
    maxno + 10, ParameterDirection.Input);
prm[1] = cmd.Parameters.Add("paramEname", OracleDbType.Varchar2,
    "Client", ParameterDirection.Input);
prm[2] = cmd.Parameters.Add("paramDeptNo", OracleDbType.Decimal,
    10, ParameterDirection.Input);
cmd.CommandText =
    "insert into emp(empno, ename, deptno) values(:1, :2, :3)";
cmd.ExecuteNonQuery();

Console.WriteLine("Record for employee id {0} has been inserted.",
    maxno + 10);
}
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Members](#)
- [OracleParameter Constructors](#)
- [OracleParameter Static Methods](#)
- [OracleParameter Properties](#)
- [OracleParameter Public Methods](#)

OracleParameter Members

OracleParameter members are listed in the following tables.

OracleParameter Constructors

OracleParameter constructors are listed in [Table 6-149](#).

Table 6-149 OracleParameter Constructors

Constructor	Description
OracleParameter Constructors	Instantiates a new instance of OracleParameter class (Overloaded)

OracleParameter Static Methods

OracleParameter static methods are listed in [Table 6-150](#).

Table 6-150 OracleParameter Static Methods

Method	Description
Equals	Inherited from System.Object (Overloaded)

OracleParameter Properties

OracleParameter properties are listed in [Table 6-151](#).

Table 6-151 OracleParameter Properties

Property	Description
ArrayBindSize	Specifies the input or output size of elements in Value property of a parameter before or after an Array Bind or PL/SQL Associative Array Bind execution
ArrayBindStatus	Specifies the input or output status of elements in Value property of a parameter before or after an Array Bind or PL/SQL Associative Array Bind execution
CollectionType	Specifies whether or not the OracleParameter represents a collection, and if so, specifies the collection type
DbType	Specifies the data type of the parameter using the Data.DbType enumeration type
Direction	Specifies whether the parameter is input-only, output-only, bi-directional, or a stored function return value parameter
IsNullable	Not supported
Offset	Specifies the offset to the Value property or offset to the elements in the Value property
OracleDbType	Specifies the Oracle data type
OracleDbTypeEx	Specifies the Oracle data type to bind the parameter as, but returns a .NET type as output
ParameterName	Specifies the name of the parameter
Precision	Specifies the maximum number of digits used to represent the Value property
Scale	Specifies the number of decimal places to which Value property is resolved
Size	Specifies the maximum size, in bytes or characters, of the data transmitted to or from the database. For PL/SQL Associative Array Bind, Size specifies the maximum number of elements in PL/SQL Associative Array
SourceColumn	Specifies the name of the DataTable Column of the DataSet
SourceColumnNullMapping	Specifies a value which indicates whether the source column is nullable
SourceVersion	Specifies the DataRowVersion value to use when loading the Value property of the parameter
Status	Indicates the status of the execution related to the data in the Value property
UdtTypeName	Specifies the Oracle user-defined type name if the parameter is a user-defined data type
Value	Specifies the value of the Parameter

OracleParameter Public Methods

OracleParameter public methods are listed in [Table 6-152](#).

Table 6-152 OracleParameter Public Methods

Public Method	Description
Clone	Creates a shallow copy of an OracleParameter object
CreateObjRef	Inherited from System.MarshalByRefObject
Dispose	Releases allocated resources
Equals	Inherited from System.Object (Overloaded)
GetHashCode	Inherited from System.Object
GetLifetimeService	Inherited from System.MarshalByRefObject
GetType	Inherited from System.Object
InitializeLifetimeService	Inherited from System.MarshalByRefObject
ResetDbType	Resets the type associated with the parameter so that it can infer its type from the value passed in the parameter
ResetOracleDbType	Resets the type associated with the parameter so that it can infer its type from the value passed in the parameter
ToString	Returns the string representation of the current instance

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)

OracleParameter Constructors

OracleParameter constructors instantiate new instances of the OracleParameter class.

Overload List:

- [OracleParameter\(\)](#)
This constructor instantiates a new instance of OracleParameter class.
- [OracleParameter\(string, OracleDbType\)](#)
This constructor instantiates a new instance of OracleParameter class using the supplied parameter name and Oracle data type.
- [OracleParameter\(string, object\)](#)
This constructor instantiates a new instance of the OracleParameter class using the supplied parameter name and parameter value.
- [OracleParameter\(string, OracleDbType, ParameterDirection\)](#)
This constructor instantiates a new instance of the OracleParameter class using the supplied parameter name, data type, and parameter direction.
- [OracleParameter\(string, OracleDbType, object, ParameterDirection\)](#)
This constructor instantiates a new instance of the OracleParameter class using the supplied parameter name, data type, value, and direction.

- [OracleParameter\(string, OracleDbType, int\)](#)

This constructor instantiates a new instance of the OracleParameter class using the supplied parameter name, data type, and size.

- [OracleParameter\(string, OracleDbType, int, string\)](#)

This constructor instantiates a new instance of the OracleParameter class using the supplied parameter name, data type, size, and source column.

- [OracleParameter\(string, OracleDbType, int, ParameterDirection, bool, byte, byte, string, DataRowVersion, object\)](#)

This constructor instantiates a new instance of the OracleParameter class using the supplied parameter name, data type, size, direction, null indicator, precision, scale, source column, source version and parameter value.

- [OracleParameter\(string, OracleDbType, int, object, ParameterDirection\)](#)

This constructor instantiates a new instance of the OracleParameter class using the supplied parameter name, data type, size, value, and direction.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)

OracleParameter()

This constructor instantiates a new instance of OracleParameter class.

Declaration

```
// C#  
public OracleParameter();
```

Remarks

Default Values:

- DbType - String
- ParameterDirection - Input
- isNullable - true
- offset - 0
- OracleDbType - Varchar2
- ParameterAlias - Empty string
- ParameterName - Empty string
- Precision - 0
- Size - 0
- SourceColumn - Empty string

- SourceVersion - Current
- ArrayBindStatus - Success
- Value - null

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)
- ["OracleParameterStatus Enumeration"](#)
- ["OracleParameterCollection Class"](#)

OracleParameter(string, OracleDbType)

This constructor instantiates a new instance of OracleParameter class using the supplied parameter name and Oracle data type.

Declaration

```
// C#  
public OracleParameter(string parameterName, OracleDbType oraType);
```

Parameters

- *parameterName*
The parameter name.
- *oraType*
The data type of the OracleParameter.

Remarks

Changing the DbType implicitly changes the OracleDbType.

Unless explicitly set in the constructor, all the properties have the default values.

Default Values:

- DbType - String
- ParameterDirection - Input
- isNullable - true
- offset - 0
- OracleDbType - Varchar2
- ParameterAlias - Empty string
- ParameterName - Empty string
- Precision - 0
- Size - 0

- SourceColumn - Empty string
- SourceVersion - Current
- ArrayBindStatus - Success
- Value - null

📘 See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleParameter Class](#)
- [OracleParameter Members](#)
- "[OracleParameterStatus Enumeration](#)"
- "[OracleParameterCollection Class](#)"

OracleParameter(string, object)

This constructor instantiates a new instance of the OracleParameter class using the supplied parameter name and parameter value.

Declaration

```
// C#  
public OracleParameter(string parameterName, object obj);
```

Parameters

- *parameterName*
The parameter name.
- *obj*
The value of the OracleParameter.

Remarks

Unless explicitly set in the constructor, all the properties have the default values.

Default Values:

- DbType - String
- ParameterDirection - Input
- isNullable - true
- offset - 0
- OracleDbType - Varchar2
- ParameterAlias - Empty string
- ParameterName - Empty string
- Precision - 0
- Size - 0

- SourceColumn - Empty string
- SourceVersion - Current
- ArrayBindStatus - Success
- Value - null

📘 See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleParameter Class](#)
- [OracleParameter Members](#)
- "[OracleParameterStatus Enumeration](#)"
- "[OracleParameterCollection Class](#)"

OracleParameter(string, OracleDbType, ParameterDirection)

This constructor instantiates a new instance of the OracleParameter class using the supplied parameter name, data type, and parameter direction.

Declaration

```
// C#  
public OracleParameter(string parameterName, OracleDbType type,  
    ParameterDirection direction);
```

Parameters

- *parameterName*
The parameter name.
- *type*
The data type of the OracleParameter.
- *direction*
The direction of the OracleParameter.

Remarks

Unless explicitly set in the constructor, all the properties have the default values.

Default Values:

- DbType - String
- ParameterDirection - Input
- isNullable - true
- offset - 0
- OracleDbType - Varchar2
- ParameterAlias - Empty string

- `ParameterName` - Empty string
- `Precision` - 0
- `Size` - 0
- `SourceColumn` - Empty string
- `SourceVersion` - Current
- `ArrayBindStatus` - Success
- `Value` - null

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)
- ["OracleParameterStatus Enumeration"](#)
- ["OracleParameterCollection Class"](#)

OracleParameter(string, OracleDbType, object, ParameterDirection)

This constructor instantiates a new instance of the `OracleParameter` class using the supplied parameter name, data type, value, and direction.

Declaration

```
// C#
public OracleParameter(string parameterName, OracleDbType type, object obj,
    ParameterDirection direction);
```

Parameters

- *parameterName*
The parameter name.
- *type*
The data type of the `OracleParameter`.
- *obj*
The value of the `OracleParameter`.
- *direction*
The `ParameterDirection` value.

Remarks

Changing the `DbType` implicitly changes the `OracleDbType`.

Unless explicitly set in the constructor, all the properties have the default values.

Default Values:

- DbType - String
- ParameterDirection - Input
- isNullable - true
- offset - 0
- OracleDbType - Varchar2
- ParameterAlias - Empty string
- ParameterName - Empty string
- Precision - 0
- Size - 0
- SourceColumn - Empty string
- SourceVersion - Current
- ArrayBindStatus - Success
- Value - null

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)
- ["OracleParameterStatus Enumeration"](#)
- ["OracleParameterCollection Class"](#)

OracleParameter(string, OracleDbType, int)

This constructor instantiates a new instance of the `OracleParameter` class using the supplied parameter name, data type, and size.

Declaration

```
// C#  
public OracleParameter(string parameterName, OracleDbType type,  
    int size);
```

Parameters

- *parameterName*
The parameter name.
- *type*
The data type of the `OracleParameter`.
- *size*
The size of the `OracleParameter` value.

Remarks

Unless explicitly set in the constructor, all the properties have the default values.

Default Values:

- DbType - String
- ParameterDirection - Input
- isNullable - true
- offset - 0
- OracleDbType - Varchar2
- ParameterAlias - Empty string
- ParameterName - Empty string
- Precision - 0
- Size - 0
- SourceColumn - Empty string
- SourceVersion - Current
- ArrayBindStatus - Success
- Value - null

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)
- ["OracleParameterStatus Enumeration"](#)
- ["OracleParameterCollection Class"](#)

OracleParameter(string, OracleDbType, int, string)

This constructor instantiates a new instance of the OracleParameter class using the supplied parameter name, data type, size, and source column.

Declaration

```
// C#
public OracleParameter(string parameterName, OracleDbType type, int size,
    string srcColumn);
```

Parameters

- *parameterName*
The parameter name.
- *type*

The data type of the `OracleParameter`.

- *size*

The size of the `OracleParameter` value.

- *srcColumn*

The name of the source column.

Remarks

Unless explicitly set in the constructor, all the properties have the default values.

Default Values:

- `DbType` - String
- `ParameterDirection` - Input
- `isNullable` - true
- `offset` - 0
- `OracleDbType` - Varchar2
- `ParameterAlias` - Empty string
- `ParameterName` - Empty string
- `Precision` - 0
- `Size` - 0
- `SourceColumn` - Empty string
- `SourceVersion` - Current
- `ArrayBindStatus` - Success
- `Value` - null

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)
- ["OracleParameterStatus Enumeration"](#)

`OracleParameter(string, OracleDbType, int, ParameterDirection, bool, byte, byte, string, DataRowVersion, object)`

This constructor instantiates a new instance of the `OracleParameter` class using the supplied parameter name, data type, size, direction, null indicator, precision, scale, source column, source version and parameter value.

Declaration

```
// C#
public OracleParameter(string parameterName, OracleDbType oraType,
```

```
int size, ParameterDirection direction, bool isNullable, byte
precision, byte scale, string srcColumn, DataRowVersion srcVersion,
object obj);
```

Parameters

- *parameterName*
The parameter name.
- *oraType*
The data type of the OracleParameter.
- *size*
The size of the OracleParameter value.
- *direction*
The ParameterDirection value.
- *isNullable*
An indicator that specifies if the parameter value can be null.
- *precision*
The precision of the parameter value.
- *scale*
The scale of the parameter value.
- *srcColumn*
The name of the source column.
- *srcVersion*
The DataRowVersion value.
- *obj*
The parameter value.

Exceptions

ArgumentException - The supplied value does not belong to the type of Value property in any of the OracleTypes.

Remarks

Unless explicitly set in the constructor, all the properties have the default values.

Default Values:

- DbType - String
- ParameterDirection - Input
- isNullable - true
- offset - 0
- OracleDbType - Varchar2
- ParameterAlias - Empty string
- ParameterName - Empty string

- Precision - 0
- Size - 0
- SourceColumn - Empty string
- SourceVersion - Current
- ArrayBindStatus - Success
- Value - null

📘 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)
- ["OracleParameterStatus Enumeration"](#)

OracleParameter(string, OracleDbType, int, object, ParameterDirection)

This constructor instantiates a new instance of the `OracleParameter` class using the supplied parameter name, data type, size, value, and direction.

Declaration

```
// C#  
public OracleParameter(string parameterName, OracleDbType type, int size,  
    object obj, ParameterDirection direction);
```

Parameters

- *parameterName*
The parameter name.
- *type*
The data type of the `OracleParameter`.
- *size*
The size of the `OracleParameter` value.
- *obj*
The value of the `OracleParameter`.
- *direction*
The `ParameterDirection` value.

Remarks

Changing the `DbType` implicitly changes the `OracleDbType`.

Unless explicitly set in the constructor, all the properties have the default values.

Default Values:

- DbType - String
- ParameterDirection - Input
- isNullable - true
- offset - 0
- OracleDbType - Varchar2
- ParameterAlias - Empty string
- ParameterName - Empty string
- Precision - 0
- Size - 0
- SourceColumn - Empty string
- SourceVersion - Current
- ArrayBindStatus - Success
- Value - null

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleParameter Class](#)
- [OracleParameter Members](#)
- "[OracleParameterStatus Enumeration](#)"
- "[OracleParameterCollection Class](#)"

OracleParameter Static Methods

The OracleParameter static method is listed in [Table 6-153](#).

Table 6-153 OracleParameter Static Method

Method	Description
Equals	Inherited from System.Object (Overloaded)

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleParameter Class](#)
- [OracleParameter Members](#)

OracleParameter Properties

OracleParameter properties are listed in [Table 6-154](#).

Table 6-154 OracleParameter Properties

Property	Description
ArrayBindSize	Specifies the input or output size of elements in Value property of a parameter before or after an Array Bind or PL/SQL Associative Array Bind execution
ArrayBindStatus	Specifies the input or output status of elements in Value property of a parameter before or after an Array Bind or PL/SQL Associative Array Bind execution
CollectionType	Specifies whether or not the OracleParameter represents a collection, and if so, specifies the collection type
DbType	Specifies the data type of the parameter using the Data.DbType enumeration type
Direction	Specifies whether the parameter is input-only, output-only, bi-directional, or a stored function return value parameter
IsNullable	Not supported
Offset	Specifies the offset to the Value property or offset to the elements in the Value property
OracleDbType	Specifies the Oracle data type
OracleDbTypeEx	Specifies the Oracle data type to bind the parameter as, but returns a .NET type as output
ParameterName	Specifies the name of the parameter
Precision	Specifies the maximum number of digits used to represent the Value property
Scale	Specifies the number of decimal places to which Value property is resolved
Size	Specifies the maximum size, in bytes or characters, of the data transmitted to or from the database. For PL/SQL Associative Array Bind, Size specifies the maximum number of elements in PL/SQL Associative Array
SourceColumn	Specifies the name of the DataTable Column of the DataSet
SourceColumnNullMapping	Specifies a value which indicates whether the source column is nullable
SourceVersion	Specifies the DataRowVersion value to use when loading the Value property of the parameter
Status	Indicates the status of the execution related to the data in the Value property
UdtTypeName	Specifies the Oracle user-defined type name if the parameter is a user-defined data type
Value	Specifies the value of the Parameter

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)

ArrayBindSize

This property specifies the maximum size, in bytes or characters, of the data for each array element transmitted to or from the database. This property is used for Array Bind or PL/SQL Associative Array execution.

Declaration

```
// C#  
public int[] ArrayBindSize {get; set; }
```

Property Value

An array of int values specifying the size.

Remarks

Default = null.

This property is only used for variable size element types for an Array Bind or PL/SQL Associative Array. For fixed size element types, this property is ignored.

Each element in the ArrayBindSize corresponds to the bind size of an element in the Value property. Before execution, ArrayBindSize specifies the maximum size of each element to be bound in the Value property. After execution, it contains the size of each element returned in the Value property.

For binding a PL/SQL Associative Array, whose elements are of a variable-length element type, as an InputOutput, Out, or ReturnValue parameter, this property must be set properly. The number of elements in ArrayBindSize must be equal to the value specified in the OracleParameter.Size property.

Example

```
// C#  
  
using System;  
using System.Data;  
using Oracle.DataAccess.Client;  
  
class ArrayBindSizeSample  
{  
    static void Main()  
    {  
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";  
        OracleConnection con = new OracleConnection(constr);  
        con.Open();  
  
        OracleParameter[] prm = new OracleParameter[3];
```

```

// Create OracleParameter objects through OracleParameterCollection
OracleCommand cmd = con.CreateCommand();

cmd.CommandText = "select max(empno) from emp";
int maxno = int.Parse(cmd.ExecuteScalar().ToString());

// Set the ArrayBindCount for Array Binding
cmd.ArrayBindCount = 2;

prm[0] = cmd.Parameters.Add("paramEmpno", OracleDbType.Decimal,
    new int[2] {maxno + 10, maxno + 11}, ParameterDirection.Input);
prm[1] = cmd.Parameters.Add("paramEname", OracleDbType.Varchar2,
    new string[2] {"Client1xxx", "Client2xxx"}, ParameterDirection.Input);
prm[2] = cmd.Parameters.Add("paramDeptNo", OracleDbType.Decimal,
    new int[2] {10, 10}, ParameterDirection.Input);

// Set the ArrayBindSize for prm[1]
// These sizes indicate the maximum size of the elements in Value property
prm[1].ArrayBindSize = new int[2];
prm[1].ArrayBindSize[0] = 7; // Set ename = "Client1"
prm[1].ArrayBindSize[1] = 7; // Set ename = "Client2"

cmd.CommandText =
    "insert into emp(empno, ename, deptno) values(:1, :2, :3)";

cmd.ExecuteNonQuery();

Console.WriteLine("Record for employee id {0} has been inserted.",
    maxno + 10);
Console.WriteLine("Record for employee id {0} has been inserted.",
    maxno + 11);

prm[0].Dispose();
prm[1].Dispose();
prm[2].Dispose();
cmd.Dispose();

con.Close();
con.Dispose();
}
}

```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)
- ["ArrayBindCount "](#)
- ["Size "](#) and ["Value "](#) for more information on binding Associative Arrays
- ["ArrayBindStatus "](#)

ArrayBindStatus

This property specifies the input or output status of each element in the Value property before or after an Array Bind or PL/SQL Associative Array execution.

Declaration

```
// C#  
public OracleParameterStatus[] ArrayBindStatus { get; set; }
```

Property Value

An array of OracleParameterStatus enumerated values.

Exceptions

ArgumentOutOfRangeException - The Status value specified is invalid.

Remarks

Default = null.

ArrayBindStatus is used for Array Bind and PL/SQL Associative Array execution only.

Before execution, ArrayBindStatus indicates the bind status of each element in the Value property. After execution, it contains the execution status of each element in the Value property.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)
- ["ArrayBindCount "](#)
- ["OracleParameterStatus Enumeration"](#)
- ["Value "](#) for more information on binding Associative Arrays
- ["ArrayBindSize "](#)

CollectionType

This property specifies whether or not the OracleParameter represents a collection, and if so, specifies the collection type.

Declaration

```
// C#  
public OracleCollectionType CollectionType { get; set; }
```

Property Value

An OracleCollectionType enumerated value.

Exceptions

ArgumentException - The OracleCollectionType value specified is invalid.

Remarks

Default = `OracleCollectionType.None`. If `OracleParameter` is used to bind a PL/SQL Associative Array, then `CollectionType` must be set to `OracleCollectionType.PLSQLAssociativeArray`.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)

DbType

This property specifies the data type of the parameter using the `Data.DbType` enumeration type.

Declaration

```
// C#  
public override DbType DbType {get; set; }
```

Property Value

A `DbType` enumerated value.

Implements

`IDataParameter`

Exceptions

`ArgumentException` - The `DbType` value specified is invalid.

Remarks

Default = `DbType.String`

`DbType` is the data type of each element in the array if the `OracleParameter` object is used for Array Bind or PL/SQL Associative Array Bind execution.

Due to the link between `DbType` and `OracleDbType` properties, if the `DbType` property is set, the `OracleDbType` property is inferred from `DbType`.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)
- ["Inference of OracleDbType from DbType"](#)
- ["CollectionType"](#)

Direction

This property specifies whether the parameter is input-only, output-only, bi-directional, or a stored function return value parameter.

Declaration

```
// C#  
public override ParameterDirection Direction { get; set; }
```

Property Value

A ParameterDirection enumerated value.

Implements

IDataParameter

Exceptions

ArgumentOutOfRangeException - The ParameterDirection value specified is invalid.

Remarks

Default = ParameterDirection.Input

Possible values: Input, InputOutput, Output, and ReturnValue.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)

IsNullable

This property is not supported.

Declaration

```
// C#  
public override bool IsNullable { get; set; }
```

Implements

IDataParameter

Property Value

This property is not supported.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)

Offset

This property specifies the offset to the Value property.

Declaration

```
// C#  
public int Offset { get; set; }
```

Property Value

An int that specifies the offset.

Exceptions

ArgumentOutOfRangeException - The Offset value specified is invalid.

Remarks

Default = 0

For Array Bind and PL/SQL Associative Array Bind, Offset applies to every element in the Value property.

The Offset property is used for binary and string data types. The Offset property represents the number of bytes for binary types and the number of characters for strings. The count for strings does not include the terminating character if a null is referenced. The Offset property is used by parameters of the following types:

- OracleDbType.BFile
- OracleDbType.Blob
- OracleDbType.LongRaw
- OracleDbType.Raw
- OracleDbType.Char
- OracleDbType.Clob
- OracleDbType.NClob

- `OracleDbType.NChar`
- `OracleDbType.NVarchar2`
- `OracleDbType.Varchar2`

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)

OracleDbType

This property specifies the Oracle data type.

Declaration

```
// C#  
public OracleDbType OracleDbType { get; set; }
```

Property Value

An `OracleDbType` enumerated value.

Remarks

Default = `OracleDbType.Varchar2`

If the `OracleParameter` object is used for Array Bind or PL/SQL Associative Array Bind execution, `OracleDbType` is the data type of each element in the array.

The `OracleDbType` property and `DbType` property are linked. Therefore, setting the `OracleDbType` property changes the `DbType` property to a supporting `DbType`.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)
- ["OracleDbType Enumeration"](#)
- ["Inference of DbType from OracleDbType"](#)
- ["CollectionType"](#)

OracleDbTypeEx

This property specifies the Oracle data type to bind the parameter as, but returns a .NET type as output.

Declaration

```
// C#  
public OracleDbType OracleDbTypeEx { get; set; }
```

Property Value

An `OracleDbType` enumerated value.

Remarks

This property is used by applications that need to bind a parameter value as an Oracle type, but need a .NET type back for output. This property should be used with an output or input/output parameter. For an input parameter, using `OracleDbTypeEx` has the same affect as using `OracleDbType`. The .NET type that is returned for the output is the .NET type that the Oracle type closely maps to.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)
- ["OracleDbType Enumeration"](#)
- ["OracleDbType "](#)
- ["Inference of DbType from OracleDbType"](#)
- ["CollectionType"](#)

ParameterName

This property specifies the name of the parameter.

Declaration

```
// C#  
public override string ParameterName { get; set; }
```

Property Value

String

Implements

`IDataParameter`

Remarks

Default = null

Oracle supports `ParameterName` up to 30 characters.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)

Precision

This property specifies the maximum number of digits used to represent the Value property.

Declaration

```
// C#  
Public byte Precision { get; set; }
```

Property Value

byte

Remarks

Default = 0

The Precision property is used by parameters of type `OracleDbType.Decimal`.

Oracle supports Precision range from 0 to 38.

For Array Bind and PL/SQL Associative Array Bind, Precision applies to each element in the Value property.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)
- ["Value"](#)

Scale

This property specifies the number of decimal places to which Value property is resolved.

Declaration

```
// C#  
public byte Scale { get; set; }
```

Property Value

byte

Remarks

Default = 0.

Scale is used by parameters of type `OracleDbType.Decimal`.

Oracle supports Scale between -84 and 127.

For Array Bind and PL/SQL Associative Array Bind, Scale applies to each element in the Value property.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)
- ["Value"](#)

Size

This property specifies the maximum size, in bytes or characters, of the data transmitted to or from the database.

Declaration

```
// C#  
public override int Size { get; set;}
```

Property Value

int

Exceptions

`ArgumentOutOfRangeException` - The Size value specified is invalid.

`InvalidOperationException` - The Size = 0 when the `OracleParameter` object is used to bind a PL/SQL Associative Array.

Remarks

If Size is not explicitly set, it is inferred from the actual size of the specified parameter value when binding only for input parameters. Output parameters must have their size defined explicitly.

The default value is 0.

Before execution, this property specifies the maximum size to be bound in the Value property. After execution, it contains the size of the type in the Value property.

Size is used for parameters of the following types:

- `OracleDbType.Blob`
- `OracleDbType.Char`

- `OracleDbType.Clob`
- `OracleDbType.LongRaw`
- `OracleDbType.NChar`
- `OracleDbType.NClob`
- `OracleDbType.NVarchar2`
- `OracleDbType.Raw`
- `OracleDbType.Varchar2`

For the preceding types, the `Size` property describes the maximum amount of data transmitted to or from the database. For character data, `Size` is in number of characters and for binary data, it is in number of bytes.

For fixed length data types, the value of `Size` is ignored.

If the `Size` is not explicitly set, it is inferred from the actual size of the specified parameter value when binding.

Note

`Size` does not include the null terminating character for the string data.

If the `OracleParameter` object is used to bind a PL/SQL Associative Array, `Size` specifies the maximum number of elements in the PL/SQL Associative Array. Before the execution, this property specifies the maximum number of elements in the PL/SQL Associative Array. After the execution, it specifies the current number of elements returned in the PL/SQL Associative Array. For `Output` and `InputOutput` parameters and return values, `Size` specifies the maximum number of elements in the PL/SQL Associative Array.

ODP.NET does not support binding an empty PL/SQL Associative Array. Therefore, `Size` cannot be set to 0 when the `OracleParameter` object is used to bind a PL/SQL Associative Array.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)
- ["OracleDbType Enumeration"](#)
- ["CollectionType"](#)
- ["ArrayBindSize"](#)
- ["ArrayBindStatus"](#)
- ["Value"](#)

SourceColumn

This property specifies the name of the DataTable Column of the DataSet.

Declaration

```
// C#  
public override string SourceColumn { get; set; }
```

Property Value

A string.

Implements

IDataParameter

Remarks

Default = empty string

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)

SourceColumnNullMapping

This property specifies a value which indicates whether the source column is nullable.

Declaration

```
// C#  
public bool SourceColumnNullMapping { get; set; }
```

Property Value

Returns true if the source column can be nullified; otherwise, returns false.

Remarks

The default value is false.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)

SourceVersion

This property specifies the DataRowVersion value to use when loading the Value property of the parameter.

Declaration

```
// C#  
public override DataRowVersion SourceVersion { get; set; }
```

Property Value

DataRowVersion

Implements

IDataParameter

Exceptions

ArgumentOutOfRangeException - The DataRowVersion value specified is invalid.

Remarks

Default = DataRowVersion.Current

SourceVersion is used by the OracleDataAdapter.UpdateCommand() during the OracleDataAdapter.Update to determine whether the original or current value is used for a parameter value. This allows primary keys to be updated. This property is ignored by the OracleDataAdapter.InsertCommand() and the OracleDataAdapter.DeleteCommand().

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)

Status

This property indicates the status of the execution related to the data in the Value property.

Declaration

```
// C#  
public OracleParameterStatus Status { get; set; }
```

Property Value

An OracleParameterStatus enumerated value.

Exceptions

ArgumentOutOfRangeException - The Status value specified is invalid.

Remarks

Default = `OracleParameterStatus.Success`

Before execution, this property indicates the bind status related to the `Value` property. After execution, it returns the status of the execution.

Status indicates if:

- A NULL is fetched from a column.
- Truncation has occurred during the fetch; then `Value` was not big enough to hold the data.
- A NULL is to be inserted into a database column; then `Value` is ignored, and a NULL is inserted into a database column.

This property is ignored for Array Bind and PL/SQL Associative Array Bind. Instead, `ArrayBindStatus` property is used.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)
- ["OracleParameterStatus Enumeration"](#)
- ["ArrayBindStatus"](#)

UdtTypeName

This property specifies the Oracle user-defined type name if the parameter is a user-defined data type.

Declaration

```
// C#  
public string UdtTypeName {get; set;}
```

Property Value

Name of the Oracle UDT.

Remarks

The `UdtTypeName` property corresponds to the user-defined type name of the parameter. This property must always be specified if the parameter is a user-defined type. Note that when a custom object is provided as an input parameter value, it is converted to the Oracle UDT that is specified by the custom type mapping on the connection used to execute the command. The Oracle UDT specified by the custom type mapping and by the `OracleParameter.UdtTypeName` property differs if the application binds a custom object that represents a subtype of the parameter type.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)

Value

This property specifies the value of the Parameter.

Declaration

```
// C#  
public override object Value { get; set; }
```

Property Value

An object.

Implements

IDataParameter

Exceptions

ArgumentException - The Value property specified is invalid.

InvalidArgumentException- The Value property specified is invalid.

Remarks

Default = null

If the OracleParameter object is used for Array Bind or PL/SQL Associative Array, Value is an array of parameter values.

The Value property can be overwritten by OracleDataAdapter.Update().

The provider attempts to convert any type of value if it supports the IConvertible interface. Conversion errors occur if the specified type is not compatible with the value.

When sending a null parameter value to the database, the user must specify DBNull, not null. The null value in the system is an empty object that has no value. DBNull is used to represent null values. The user can also specify a null value by setting Status to OracleParameterStatus.NullValue. In this case, the provider sends a null value to the database.

If neither OracleDbType nor DbType are set, their values can be inferred by Value. Please see the following for related information:

- Tables in section ["Inference of DbType and OracleDbType from Value"](#)
- ["ArrayBindCount "](#)
- ["ArrayBindSize "](#)
- ["ArrayBindStatus "](#)

- ["OracleDbType Enumeration"](#)

For input parameters the value is:

- Bound to the `OracleCommand` that is sent to the database.
- Converted to the data type specified in `OracleDbType` or `DbType` when the provider sends the data to the database.

For output parameters the value is:

- Set on completion of the `OracleCommand` (true for return value parameters also).
- Set to the data from the database, to the data type specified in `OracleDbType` or `DbType`.

When array binding is used with:

- Input parameter - Value should be set to an array of values. `OracleCommand.ArrayBindCount` should be set to a value that is greater than zero to indicate the number of elements to be bound.

The number of elements in the array should be equal to the `OracleCommand.ArrayBindCount` property; otherwise, their minimum value is used to bind the elements in the array.
- Output parameter - `OracleCommand.ArrayBindCount` should be set to a value that is greater than zero to indicate the number of elements to be retrieved (for SELECT statements).

When PL/SQL Associative Array binding is used with:

- Input parameter – Value should be set to an array of values. `CollectionType` should be set to `OracleCollection.PLSQLAssociativeArray`. `Size` should be set to specify the possible maximum number of array elements in the PL/SQL Associative Array. If `Size` is smaller than the number of elements in `Value`, then `Size` specifies the number of elements in the `Value` property to be bound.
- Output parameter - `CollectionType` should be set to `OracleCollection.PLSQLAssociativeArray`. `Size` should be set to specify the maximum number of array elements in PL/SQL Associative Array.

Each parameter should have a value. To bind a parameter with a null value, set `Value` to `DBNull.Value`, or set `Status` to `OracleParameterStatus.NullInsert`.

If the data type specified is `OracleDbType.Decimal`, precision is not set or set to 0, and `OracleConfiguration` or `OracleConnection.SuppressGetDecimalInvalidCastException` is set to true, then a rounded-off 28 or 29 precision Oracle NUMBER value is return that can be represented as a .NET Decimal.

For vectors, if `OracleParameter.Value` is a .NET numeric type, then ODP.NET converts the value to the numeric format specified in the bind type (`OracleDbType.Vector*`) in `OracleParameter` unless it is a BINARY vector. In the BINARY case, the parameter values must be a .NET byte[] or string type, such as .NET string or `OracleString`.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)
- ["ArrayBindCount "](#)
- ["OracleParameterStatus Enumeration"](#)

OracleParameter Public Methods

OracleParameter public methods are listed in [Table 6-155](#).

Table 6-155 OracleParameter Public Methods

Public Method	Description
Clone	Creates a shallow copy of an OracleParameter object
CreateObjRef	Inherited from System.MarshalByRefObject
Dispose	Releases allocated resources
Equals	Inherited from System.Object (Overloaded)
GetHashCode	Inherited from System.Object
GetLifetimeService	Inherited from System.MarshalByRefObject
GetType	Inherited from System.Object
InitializeLifetimeService	Inherited from System.MarshalByRefObject
ResetDbType	Resets the type associated with the parameter so that it can infer its type from the value passed in the parameter
ResetOracleDbType	Resets the type associated with the parameter so that it can infer its type from the value passed in the parameter
ToString	Returns the string representation of the current instance

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)

Clone

This method creates a shallow copy of an OracleParameter object.

Declaration

```
// C#  
public object Clone();
```

Return Value

An OracleParameter object.

Implements

ICloneable

Remarks

The cloned object has the same property values as that of the object being cloned.

Example

```
// C#  
  
using System;  
using System.Data;  
using Oracle.DataAccess.Client;  
  
class CloneSample  
{  
    static void Main()  
    {  
        OracleParameter prm1 = new OracleParameter();  
  
        // Prints "prm1.ParameterName = "  
        Console.WriteLine("prm1.ParameterName = " + prm1.ParameterName);  
  
        // Set the ParameterName before cloning  
        prm1.ParameterName = "MyParam";  
  
        // Clone the OracleParameter  
        OracleParameter prm2 = (OracleParameter) prm1.Clone();  
  
        // Prints "prm2.ParameterName = MyParam"  
        Console.WriteLine("prm2.ParameterName = " + prm2.ParameterName);  
  
        prm1.Dispose();  
        prm2.Dispose();  
    }  
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)

Dispose

This method releases resources allocated for an OracleParameter object.

Declaration

```
// C#  
public void Dispose();
```

Implements

IDisposable

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)

ResetDbType

This method resets the type associated with the parameter so that it can infer its type from the value passed in the parameter.

Declaration

```
// C#  
public override void ResetDbType();
```

Remarks

If an application does not set the DbType or OracleDbType properties of an OracleParameter object, then these values are inferred from the value set by the application to that OracleParameter object. Calling ResetDbType method resets these properties so that OracleParameter can again infer its type from the value passed into the OracleParameter. Calling this method affects both the DbType and OracleDbType properties of the OracleParameter object.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)

ResetOracleDbType

This method resets the type associated with the parameter so that it can infer its type from the value passed in the parameter.

Declaration

```
// C#  
public override void ResetOracleDbType();
```

Remarks

If an application does not set the `DbType` or `OracleDbType` properties of an `OracleParameter` object, then these values are inferred from the value set by the application to that `OracleParameter` object. Calling the `ResetOracleDbType` method resets these properties so that `OracleParameter` can again infer its type from the value passed into the `OracleParameter`. Calling this method affects both the `DbType` and `OracleDbType` properties of the `OracleParameter` object.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)

ToString

Overrides `Object`

This method returns the string representation of the current instance.

Declaration

```
// C#  
public override string ToString();
```

Return Value

Returns the `OracleParameter` value in a string representation.

Remarks

If the current instance has a null value, the returned string is null.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameter Class](#)
- [OracleParameter Members](#)

OracleParameterCollection Class

An `OracleParameterCollection` class represents a collection of all parameters relevant to an `OracleCommand` object and their mappings to `DataSet` columns.

Class Inheritance

System.Object
 System.MarshalByRefObject
 System.Data.Common.DbParameterCollection
 Oracle.DataAccess.Client.OracleParameterCollection

Declaration

```
// C#
public sealed class OracleParameterCollection : DbParameterCollection,
    IDataParameterCollection, IList, ICollection, IEnumerable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

The position of an OracleParameter added into the OracleParameterCollection is the binding position in the SQL statement. Position is 0-based and is used only for positional binding. If named binding is used, the position of an OracleParameter in the OracleParameterCollection is ignored.

Example

```
// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;

class OracleParameterCollectionSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleParameter[] prm = new OracleParameter[3];
```

```

// Create OracleParameter objects through OracleParameterCollection
OracleCommand cmd = con.CreateCommand();

cmd.CommandText = "select max(empno) from emp";
int maxno = int.Parse(cmd.ExecuteScalar().ToString());

prm[0] = cmd.Parameters.Add("paramEmpno", OracleDbType.Decimal,
    maxno + 10, ParameterDirection.Input);
prm[1] = cmd.Parameters.Add("paramEname", OracleDbType.Varchar2,
    "Client", ParameterDirection.Input);
prm[2] = cmd.Parameters.Add("paramDeptNo", OracleDbType.Decimal,
    10, ParameterDirection.Input);
cmd.CommandText =
    "insert into emp(empno, ename, deptno) values(:1, :2, :3)";
cmd.ExecuteNonQuery();

Console.WriteLine("Record for employee id {0} has been inserted.",
    maxno + 10);

// Remove all parameters from OracleParameterCollection
cmd.Parameters.Clear();

prm[0].Dispose();
prm[1].Dispose();
prm[2].Dispose();
cmd.Dispose();

con.Close();
con.Dispose();
}
}

```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Members](#)
- [OracleParameterCollection Static Methods](#)
- [OracleParameterCollection Properties](#)
- [OracleParameterCollection Public Methods](#)

OracleParameterCollection Members

OracleParameterCollection members are listed in the following tables.

OracleParameterCollection Static Methods

OracleParameterCollection static methods are listed in [Table 6-156](#).

Table 6-156 OracleParameterCollection Static Methods

Method	Description
Equals	Inherited from System.Object (Overloaded)

OracleParameterCollection Properties

OracleParameterCollection properties are listed in [Table 6-157](#).

Table 6-157 OracleParameterCollection Properties

Property	Description
Count	Specifies the number of OracleParameters in the collection
Item	Gets and sets the OracleParameter object (Overloaded)
IsFixedSize	Gets a value that indicates whether the OracleParameterCollection has a fixed size
IsReadOnly	Gets a value that indicates whether the OracleParameterCollection is read-only
IsSynchronized	Gets a value that indicates whether the OracleParameterCollection is synchronized.
SyncRoot	Gets an object that can be used to synchronize access to the OracleParameterCollection

OracleParameterCollection Public Methods

OracleParameterCollection public methods are listed in [Table 6-158](#).

Table 6-158 OracleParameterCollection Public Methods

Public Method	Description
Add	Adds objects to the collection (Overloaded)
AddRange	Adds elements to the end of the OracleParameterCollection
Clear	Removes all the OracleParameter objects from the collection
Contains	Indicates whether or not objects exist in the collection (Overloaded)
CopyTo	Copies OracleParameter objects from the collection, starting with the supplied index to the supplied array
CreateObjRef	Inherited from System.MarshalByRefObject
Equals	Inherited from System.Object (Overloaded)
GetEnumerator	Returns an enumerator that iterates through the OracleParameterCollection
GetHashCode	Inherited from System.Object
GetLifetimeService	Inherited from System.MarshalByRefObject
GetType	Inherited from System.Object
InitializeLifetimeService	Inherited from System.MarshalByRefObject
IndexOf	Returns the index of the objects in the collection (Overloaded)
Insert	Inserts the supplied OracleParameter to the collection at the specified index

Table 6-158 (Cont.) OracleParameterCollection Public Methods

Public Method	Description
Remove	Removes objects from the collection
RemoveAt	Removes objects from the collection by location (Overloaded)
ToString	Inherited from System.Object

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleParameterCollection Class](#)

OracleParameterCollection Static Methods

The OracleParameterCollection static method is listed in [Table 6-159](#).

Table 6-159 OracleParameterCollection Static Method

Method	Description
Equals	Inherited from System.Object (Overloaded)

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

OracleParameterCollection Properties

OracleParameterCollection properties are listed in [Table 6-160](#).

Table 6-160 OracleParameterCollection Properties

Property	Description
Count	Specifies the number of OracleParameters in the collection
Item	Gets and sets the OracleParameter object (Overloaded)
IsFixedSize	Gets a value that indicates whether the OracleParameterCollection has a fixed size
IsReadOnly	Gets a value that indicates whether the OracleParameterCollection is read-only

Table 6-160 (Cont.) OracleParameterCollection Properties

Property	Description
IsSynchronized	Gets a value that indicates whether the OracleParameterCollection is synchronized.
SyncRoot	Gets an object that can be used to synchronize access to the OracleParameterCollection

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

Count

This property specifies the number of OracleParameter objects in the collection.

Declaration

```
// C#  
public override int Count {get;}
```

Property Value

The number of OracleParameter objects.

Implements

ICollection

Remarks

Default = 0

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

Item

Item gets and sets the OracleParameter object.

Overload List:

- [Item\[int\]](#)
This property gets and sets the OracleParameter object at the index specified by the supplied parameterIndex.
- [Item\[string\]](#)
This property gets and sets the OracleParameter object using the parameter name specified by the supplied parameterName.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

Item[int]

This property gets and sets the OracleParameter object at the index specified by the supplied parameterIndex.

Declaration

```
// C#  
public object Item[int parameterIndex] {get; set;}
```

Property Value

An object.

Implements

IList

Exceptions

IndexOutOfRangeException - The supplied index does not exist.

Remarks

The OracleParameterCollection class is a zero-based index.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

Item[string]

This property gets and sets the `OracleParameter` object using the parameter name specified by the supplied `parameterName`.

Declaration

```
// C#  
public OracleParameter Item[string parameterName] {get; set;};
```

Property Value

An `OracleParameter`.

Implements

`IDataParameterCollection`

Exceptions

`IndexOutOfRangeException` - The supplied parameter name does not exist.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

IsFixedSize

`IsFixedSize` gets a value that indicates whether the `OracleParameterCollection` has a fixed size.

Declaration

```
// C#  
public override bool IsFixedSize { get; };
```

Property Value

Returns true if the `OracleParameterCollection` has a fixed size; otherwise false.

Implements

`IList`

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

IsReadOnly

IsReadOnly gets a value that indicates whether the OracleParameterCollection is read-only.

Declaration

```
// C#  
public override bool IsReadOnly { get; };
```

Property Value

Returns true if the OracleParameterCollection is read only; otherwise false.

Implements

ICollection

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

IsSynchronized

IsSynchronized gets a value that indicates whether the OracleParameterCollection is synchronized.

Declaration

```
// C#  
public override bool IsSynchronized { get; };
```

Property Value

Returns true if the OracleParameterCollection is synchronized; otherwise false.

Implements

ICollection

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

SyncRoot

SyncRoot gets an object that can be used to synchronize access to the OracleParameterCollection.

Declaration

```
// C#
public override Object SyncRoot { get; };
```

Property Value

An object that can be used to synchronize access to the OracleParameterCollection.

Implements

ICollection

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

OracleParameterCollection Public Methods

OracleParameterCollection public methods are listed in [Table 6-161](#).

Table 6-161 OracleParameterCollection Public Methods

Public Method	Description
Add	Adds objects to the collection (Overloaded)
AddRange	Adds elements to the end of the OracleParameterCollection
Clear	Removes all the OracleParameter objects from the collection
Contains	Indicates whether or not objects exist in the collection (Overloaded)
CopyTo	Copies OracleParameter objects from the collection, starting with the supplied index to the supplied array

Table 6-161 (Cont.) OracleParameterCollection Public Methods

Public Method	Description
CreateObjRef	Inherited from System.MarshalByRefObject
Equals	Inherited from System.Object (Overloaded)
GetEnumerator	Returns an enumerator that iterates through the OracleParameterCollection
GetHashCode	Inherited from System.Object
GetLifetimeService	Inherited from System.MarshalByRefObject
GetType	Inherited from System.Object
InitializeLifetimeService	Inherited from System.MarshalByRefObject
IndexOf	Returns the index of the objects in the collection (Overloaded)
Insert	Inserts the supplied OracleParameter to the collection at the specified index
Remove	Removes objects from the collection
RemoveAt	Removes objects from the collection by location (Overloaded)
ToString	Inherited from System.Object

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

Add

Add adds objects to the collection.

Overload List:

- [Add\(object\)](#)
This method adds the supplied object to the collection.
- [Add\(OracleParameter\)](#)
This method adds the supplied OracleParameter object to the collection.
- [Add\(string, object\)](#)
This method adds an OracleParameter object to the collection using the supplied name and object value.
- [Add\(string, OracleDbType\)](#)
This method adds an OracleParameter object to the collection using the supplied name and database type.

- [Add\(string, OracleDbType, ParameterDirection\)](#)
This method adds an `OracleParameter` object to the collection using the supplied name, database type, and direction.
- [Add\(string, OracleDbType, object, ParameterDirection\)](#)
This method adds an `OracleParameter` object to the collection using the supplied name, database type, parameter value, and direction.
- [Add\(string, OracleDbType, int, object, ParameterDirection\)](#)
This method adds an `OracleParameter` object to the collection using the supplied name, database type, size, parameter value, and direction.
- [Add\(string, OracleDbType, int\)](#)
This method adds an `OracleParameter` object to the collection using the supplied name, database type, and size.
- [Add \(string, OracleDbType, int, string\)](#)
This method adds an `OracleParameter` object to the collection using the supplied name, database type, size, and source column.
- [Add\(string, OracleDbType, int, ParameterDirection, bool, byte, byte, string, DataRowVersion, object\)](#)
This method adds an `OracleParameter` object to the collection using the supplied name, database type, size, direction, null indicator, precision, scale, source column, source version, and parameter value.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

Add(object)

This method adds the supplied object to the collection.

Declaration

```
// C#  
public override int Add(object obj);
```

Parameters

- *obj*
The supplied object.

Return Value

The index at which the new `OracleParameter` is added.

Implements

ICollection

Remarks

InvalidOperationException - The supplied *obj* cannot be cast to an OracleParameter object.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

Add(OracleParameter)

This method adds the supplied OracleParameter object to the collection.

Declaration

```
// C#  
public OracleParameter Add(OracleParameter paramObj);
```

Parameters

- *paramObj*
The supplied OracleParameter object.

Return Value

The newly created OracleParameter object which was added to the collection.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

Add(string, object)

This method adds an OracleParameter object to the collection using the supplied name and object value

Declaration

```
// C#  
public OracleParameter Add(string name, object val);
```

Parameters

- *name*
The parameter name.
- *val*
The OracleParameter value.

Return Value

The newly created OracleParameter object which was added to the collection.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

Add(string, OracleDbType)

This method adds an OracleParameter object to the collection using the supplied name and database type.

Declaration

```
// C#  
public OracleParameter Add(string name, OracleDbType dbType);
```

Parameters

- *name*
The parameter name.
- *dbType*
The data type of the OracleParameter.

Return Value

The newly created OracleParameter object which was added to the collection.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

Add(string, OracleDbType, ParameterDirection)

This method adds an `OracleParameter` object to the collection using the supplied name, database type, and direction.

Declaration

```
// C#  
public OracleParameter Add(string name, OracleDbType dbType,  
    ParameterDirection direction);
```

Parameters

- *name*
The parameter name.
- *dbType*
The data type of the `OracleParameter`.
- *direction*
The `OracleParameter` direction.

Return Value

The newly created `OracleParameter` object which was added to the collection.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)
- ["OracleDbType Enumeration"](#)

Add(string, OracleDbType, object, ParameterDirection)

This method adds an `OracleParameter` object to the collection using the supplied name, database type, parameter value, and direction.

Declaration

```
// C#  
public OracleParameter Add(string name, OracleDbType dbType, object val,  
    ParameterDirection dir);
```

Parameters

- *name*
The parameter name.
- *dbType*
The data type of the `OracleParameter`.

- *val*
The OracleParameter value.
- *dir*
The ParameterDirection value.

Return Value

The newly created OracleParameter object which was added to the collection.

Example

```
// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;

class AddSample
{
    static void Main()
    {
        OracleCommand cmd = new OracleCommand();

        // Add parameter to the OracleParameterCollection
        OracleParameter prm = cmd.Parameters.Add(
            "MyParam", OracleDbType.Decimal, 1, ParameterDirection.Input);

        // Prints "cmd.Parameters.Count = 1"
        Console.WriteLine("cmd.Parameters.Count = " + cmd.Parameters.Count);

        prm.Dispose();
        cmd.Dispose();
    }
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)
- ["OracleDbType Enumeration"](#)

Add(string, OracleDbType, int, object, ParameterDirection)

This method adds an OracleParameter object to the collection using the supplied name, database type, size, parameter value, and direction.

Declaration

```
// C#
public OracleParameter Add(string name, OracleDbType dbType, int size,
    object val, ParameterDirection dir;
```

Parameters

- *name*
The parameter name.
- *dbType*
The data type of the OracleParameter.
- *size*
The size of OracleParameter.
- *val*
The OracleParameter value.
- *dir*
The ParameterDirection value.

Return Value

The newly created OracleParameter object which was added to the collection.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)
- ["OracleDbType Enumeration"](#)

Add(string, OracleDbType, int)

This method adds an OracleParameter object to the collection using the supplied name, database type, and size.

Declaration

```
// C#  
public OracleParameter Add(string name, OracleDbType dbType, int size);
```

Parameters

- *name*
The parameter name.
- *dbType*
The data type of the OracleParameter.
- *size*
The size of OracleParameter.

Return Value

The newly created `OracleParameter` object which was added to the collection.

Example

```
// C#

using System;
using Oracle.DataAccess.Client;

class AddSample
{
    static void Main()
    {
        OracleCommand cmd = new OracleCommand();

        // Add parameter to the OracleParameterCollection
        OracleParameter prm = cmd.Parameters.Add(
            "MyParam", OracleDbType.Varchar2, 10);

        // Prints "cmd.Parameters.Count = 1"
        Console.WriteLine("cmd.Parameters.Count = " + cmd.Parameters.Count);

        prm.Dispose();
        cmd.Dispose();
    }
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

Add (string, OracleDbType, int, string)

This method adds an `OracleParameter` object to the collection using the supplied name, database type, size, and source column.

Declaration

```
// C#
public OracleParameter Add(string name, OracleDbType dbType, int size,
    string srcColumn);
```

Parameters

- *name*
The parameter name.
- *dbType*
The data type of the `OracleParameter`.
- *size*

The size of `OracleParameter`.

- *srcColumn*

The name of the source column.

Return Value

An `OracleParameter`.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

Add(string, OracleDbType, int, ParameterDirection, bool, byte, byte, string, DataRowVersion, object)

This method adds an `OracleParameter` object to the collection using the supplied name, database type, size, direction, null indicator, precision, scale, source column, source version, and parameter value.

Declaration

```
// C#
public OracleParameter Add(string name, OracleDbType dbType, int size,
    ParameterDirection dir, bool isNullable, byte precision,
    byte scale, string srcColumn, DataRowVersion version, object val);
```

Parameters

- *name*
The parameter name.
- *dbType*
The data type of the `OracleParameter`.
- *size*
The size of `OracleParameter`.
- *dir*
The `ParameterDirection` value.
- *isNullable*
An indicator that specifies if the parameter value can be null.
- *precision*
The precision of the parameter value.
- *scale*
The scale of the parameter value.

- *srcColumn*
The name of the source column.
- *version*
The DataRowVersion value.
- *val*
The parameter value.

Return Value

The newly created OracleParameter object which was added to the collection.

Exceptions

ArgumentException - The type of supplied *val* does not belong to the type of Value property in any of the ODP.NET Types.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

AddRange

This method adds elements to the end of the OracleParameterCollection.

Declaration

```
// C#  
public override void AddRange(Array paramArray );
```

Parameters

paramArray

An array of OracleParameter objects.

Exceptions

ArgumentNullException - The input parameter is null.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

Clear

This method removes all the OracleParameter objects from the collection.

Declaration

```
// C#  
public override void Clear();
```

Implements

ICollection

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Client;  
  
class ClearSample  
{  
    static void Main()  
    {  
        OracleCommand cmd = new OracleCommand();  
  
        // Add parameter to the OracleParameterCollection  
        OracleParameter prm = cmd.Parameters.Add("MyParam", OracleDbType.Decimal);  
  
        // Prints "cmd.Parameters.Count = 1"  
        Console.WriteLine("cmd.Parameters.Count = " + cmd.Parameters.Count);  
  
        // Clear all parameters in the OracleParameterCollection  
        cmd.Parameters.Clear();  
  
        // Prints "cmd.Parameters.Count = 0"  
        Console.WriteLine("cmd.Parameters.Count = " + cmd.Parameters.Count);  
  
        prm.Dispose();  
        cmd.Dispose();  
    }  
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

Contains

Contains indicates whether or not the supplied object exists in the collection.

Overload List:

- [Contains\(object\)](#)
This method indicates whether or not the supplied object exists in the collection.
- [Contains\(string\)](#)
This method indicates whether or not an OracleParameter object exists in the collection using the supplied string.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

Contains(object)

This method indicates whether or not the supplied object exists in the collection.

Declaration

```
// C#  
public override bool Contains(object obj)
```

Parameters

- *obj*
The object.

Return Value

A bool that indicates whether or not the OracleParameter specified is inside the collection.

Implements

ICollection

Exceptions

InvalidCastException - The supplied *obj* is not an OracleParameter object.

Remarks

Returns true if the collection contains the OracleParameter object; otherwise, returns false.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Client;  
  
class ContainsSample  
{
```

```
static void Main()
{
    OracleCommand cmd = new OracleCommand();

    // Add parameter to the OracleParameterCollection
    OracleParameter prm1 = cmd.Parameters.Add("MyParam", OracleDbType.Decimal);

    // Check if the OracleParameterCollection contains prm1
    bool bContains = cmd.Parameters.Contains(prm1);

    // Prints "bContains = True"
    Console.WriteLine("bContains = " + bContains);

    OracleParameter prm2 = new OracleParameter();

    // Check if the OracleParameterCollection contains prm2
    bContains = cmd.Parameters.Contains(prm2);

    // Prints "bContains = False"
    Console.WriteLine("bContains = " + bContains);

    prm1.Dispose();
    prm2.Dispose();
    cmd.Dispose();
}
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

Contains(string)

This method indicates whether or not an `OracleParameter` object exists in the collection using the supplied string.

Declaration

```
// C#
public override bool Contains(string name);
```

Parameters

- *name*
The name of `OracleParameter` object.

Return Value

Returns `true` if the collection contains the `OracleParameter` object with the specified parameter name; otherwise, returns `false`.

Implements

`IDataParameterCollection`

Example

```
// C#

using System;
using Oracle.DataAccess.Client;

class ContainsSample
{
    static void Main()
    {
        OracleCommand cmd = new OracleCommand();

        // Add parameter to the OracleParameterCollection
        OracleParameter prm = cmd.Parameters.Add("MyParam", OracleDbType.Decimal);

        // Check if the OracleParameterCollection contains "MyParam"
        bool bContains = cmd.Parameters.Contains("MyParam");

        // Prints "bContains = True"
        Console.WriteLine("bContains = " + bContains);

        // Check if the OracleParameterCollection contains "NoParam"
        bContains = cmd.Parameters.Contains("NoParam");

        // Prints "bContains = False"
        Console.WriteLine("bContains = " + bContains);

        prm.Dispose();
        cmd.Dispose();
    }
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

CopyTo

This method copies `OracleParameter` objects from the collection, starting with the supplied index to the supplied array.

Declaration

```
// C#
public override void CopyTo(Array array, int index);
```

Parameters

- *array*
The specified array.
- *index*

The array index.

Implements

ICollection

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

GetEnumerator

GetEnumerator returns an enumerator that iterates through the OracleParameterCollection.

Declaration

```
// C#  
public override IEnumerator GetEnumerator();
```

Implements

IEnumerable

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

IndexOf

IndexOf returns the index of the OracleParameter object in the collection.

Overload List:

- [IndexOf\(object\)](#)
This method returns the index of the OracleParameter object in the collection.
- [IndexOf\(String\)](#)
This method returns the index of the OracleParameter object with the specified name in the collection.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

IndexOf(object)

This method returns the index of the `OracleParameter` object in the collection.

Declaration

```
// C#  
public override int IndexOf(object obj);
```

Parameters

- *obj*

The specified object.

Return Value

Returns the index of the `OracleParameter` object in the collection.

Implements

`IList`

Exceptions

`InvalidCastException` - The supplied *obj* cannot be cast to an `OracleParameter` object.

Remarks

Returns the index of the supplied `OracleParameter` *obj* in the collection.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

IndexOf(String)

This method returns the index of the `OracleParameter` object with the specified name in the collection.

Declaration

```
// C#  
public override int IndexOf(String name);
```

Parameters

- *name*

The name of parameter.

Return Value

Returns the index of the supplied `OracleParameter` in the collection.

Implements

`IDataParameterCollection`

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

Insert

This method inserts the supplied `OracleParameter` object to the collection at the specified index.

Declaration

```
// C#  
public override void Insert(int index, object obj);
```

Parameters

- *index*

The specified index.

- *obj*

The `OracleParameter` object.

Implements

`IList`

Remarks

An `InvalidCastException` is thrown if the supplied *obj* cannot be cast to an `OracleParameter` object.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

Remove

This method removes the supplied `OracleParameter` from the collection.

Declaration

```
// C#  
public override void Remove(object obj);
```

Parameters

- *obj*
The specified object to remove.

Implements

`IList`

Exceptions

`InvalidCastException` - The supplied *obj* cannot be cast to an `OracleParameter` object.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Client;  
  
class RemoveSample  
{  
    static void Main()  
    {  
        OracleCommand cmd = new OracleCommand();  
  
        // Add 2 parameters to the OracleParameterCollection  
        OracleParameter prm1 = cmd.Parameters.Add("MyParam1", OracleDbType.Decimal);  
        OracleParameter prm2 = cmd.Parameters.Add("MyParam2", OracleDbType.Decimal);  
  
        // Prints "cmd.Parameters.Count = 2"  
        Console.WriteLine("cmd.Parameters.Count = " + cmd.Parameters.Count);  
  
        // Remove the 1st parameter from the OracleParameterCollection  
        cmd.Parameters.Remove(prm1);  
  
        // Prints "cmd.Parameters.Count = 1"  
        Console.WriteLine("cmd.Parameters.Count = " + cmd.Parameters.Count);  
  
        // Prints "cmd.Parameters[0].ParameterName = MyParam2"  
        Console.WriteLine("cmd.Parameters[0].ParameterName = " +
```



```
cmd.Parameters[0].ParameterName);  
  
prm1.Dispose();  
prm2.Dispose();  
cmd.Dispose();  
}  
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

RemoveAt

RemoveAt removes the OracleParameter object from the collection by location.

Overload List:

- [RemoveAt\(int\)](#)
This method removes from the collection the OracleParameter object located at the index specified by the supplied index.
- [RemoveAt\(String\)](#)
This method removes from the collection the OracleParameter object specified by the supplied name.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

RemoveAt(int)

This method removes from the collection the OracleParameter object located at the index specified by the supplied index.

Declaration

```
// C#  
public override void RemoveAt(int index);
```

Parameters

- *index*
The specified index from which the OracleParameter is to be removed.

Implements

IList

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

RemoveAt(String)

This method removes from the collection the `OracleParameter` object specified by the supplied name.

Declaration

```
// C#  
public override void RemoveAt(String name);
```

Parameters

- *name*

The name of the `OracleParameter` object to be removed from the collection.

Implements

IDataParameterCollection

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleParameterCollection Class](#)
- [OracleParameterCollection Members](#)

OraclePermission Class

An `OraclePermission` object enables ODP.NET to enforce imperative security and helps ensure that a user has a security level adequate for accessing data.

Class Inheritance

System.Object

System.Security.CodeAccessPermission

System.Data.Common.DbDataPermission

Oracle.DataAccess.Client.OraclePermission

Declaration

```
// C#
public class OraclePermission: DBDataPermission
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OraclePermission Members](#)
- [OraclePermission Constructor](#)
- [OraclePermission Static Methods](#)
- [OraclePermission Public Properties](#)
- [OraclePermission Public Methods](#)

OraclePermission Members

OraclePermission members are listed in the following tables.

OraclePermission Constructors

The OraclePermission constructor is listed in [Table 6-162](#).

Table 6-162 OraclePermission Constructor

Constructor	Description
OraclePermission Constructor	Instantiates a new instance of the OraclePermission class.

OraclePermission Static Methods

The OraclePermission static methods are listed in [Table 6-163](#).

Table 6-163 OraclePermission Static Methods

Static Method	Description
Equals	Inherited from System.Object
ReferenceEquals	Inherited from System.Object
RevertAll	Inherited from CodeAccessPermission
RevertAssert	Inherited from CodeAccessPermission
RevertDeny	Inherited from CodeAccessPermission
RevertPermitOnly	Inherited from CodeAccessPermission

OraclePermission Public Properties

The OraclePermission public methods are listed in [Table 6-167](#).

Table 6-164 OraclePermission Public Properties

Public Properties	Description
AllowBlankPassword	Inherited from DBDataPermission OraclePermission does not support this property.

OraclePermission Public Methods

The OraclePermission public methods are listed in [Table 6-165](#).

Table 6-165 OraclePermission Public Methods

Public Method	Description
Add	Adds a new connection string fragment and a list of restricted keywords to the OraclePermission object
Assert	Inherited from CodeAccessPermission
Copy	Returns a copy of the current permission object
Demand	Inherited from CodeAccessPermission
Deny	Inherited from CodeAccessPermission
Equals	Inherited from CodeAccessPermission
FromXml	Inherited from DBDataPermission
GetHashCode	Inherited from CodeAccessPermission
GetType	Inherited from System.Object
Intersect	Inherited from DBDataPermission
IsSubsetOf	Returns a boolean value that indicates whether or not the current permission is a subset of the target permission
IsUnrestricted	Inherited from DBDataPermission
PermitOnly	Inherited from CodeAccessPermission
ToString	Inherited from CodeAccessPermission
ToXml	Inherited from DBDataPermission

Table 6-165 (Cont.) OraclePermission Public Methods

Public Method	Description
Union	Inherited from DBDataPermission

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OraclePermission Class](#)

OraclePermission Constructor

The OraclePermission constructor instantiates a new instance of the OraclePermission class.

Declaration

```
// C#  
public OraclePermission (PermissionState state);
```

Parameters

- *state*

The *state* parameter takes one of the following two values: `PermissionState.None` or `PermissionState.Unrestricted`.

Exceptions

`ArgumentException` - The `PermissionState` value is invalid.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OraclePermission Class](#)
- [OraclePermission Members](#)

OraclePermission Static Methods

The OraclePermission static methods are listed in [Table 6-166](#).

Table 6-166 OraclePermission Static Methods

Static Method	Description
Equals	Inherited from <code>System.Object</code>
ReferenceEquals	Inherited from <code>System.Object</code>

Table 6-166 (Cont.) OraclePermission Static Methods

Static Method	Description
RevertAll	Inherited from CodeAccessPermission
RevertAssert	Inherited from CodeAccessPermission
RevertDeny	Inherited from CodeAccessPermission
RevertPermitOnly	Inherited from CodeAccessPermission

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OraclePermission Class](#)
- [OraclePermission Members](#)

OraclePermission Public Properties

The OraclePermission public methods are listed in [Table 6-167](#).

Table 6-167 OraclePermission Public Properties

Public Properties	Description
AllowBlankPassword	Inherited from DBDataPermission OraclePermission ignores the value of this property. Any value set for this property, for an OraclePermission object, is ignored.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OraclePermission Class](#)
- [OraclePermission Members](#)

OraclePermission Public Methods

The OraclePermission public methods are listed in [Table 6-168](#).

Table 6-168 OraclePermission Public Methods

Public Method	Description
Add	Adds a new connection string fragment and a list of restricted keywords to the OraclePermission object

Table 6-168 (Cont.) OraclePermission Public Methods

Public Method	Description
Assert	Inherited from CodeAccessPermission
Copy	Returns a copy of the current permission object
Demand	Inherited from CodeAccessPermission
Deny	Inherited from CodeAccessPermission
Equals	Inherited from CodeAccessPermission
FromXml	Inherited from DBDataPermission
GetHashCode	Inherited from CodeAccessPermission
GetType	Inherited from System.Object
Intersect	Inherited from DBDataPermission
IsSubsetOf	Returns a boolean value that indicates whether or not the current permission is a subset of the target permission
IsUnrestricted	Inherited from DBDataPermission
PermitOnly	Inherited from CodeAccessPermission
ToString	Inherited from CodeAccessPermission
ToXml	Inherited from DBDataPermission
Union	Inherited from DBDataPermission

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OraclePermission Class](#)
- [OraclePermission Members](#)

Add

This method adds a new connection string fragment and a list of restricted keywords to the OraclePermission object.

Declaration

```
// C#
public void Add(string connStr, string keyRestrict,
    KeyRestrictionBehavior behavior);
```

Parameters

- *connStr*
The connection string fragment.
- *keyRestrict*
The key restrictions.

- *behavior*

One of the following KeyRestrictionBehavior enumerations:

- AllowOnly
- PreventUsage

Exceptions

ArgumentException - The KeyRestrictionBehavior value or the format of the *connStr* or *keyRestrict* string is invalid.

Remarks

The Add method configures the connection strings allowed or disallowed by the permission object.

Opening an OracleConnection is allowed or denied based upon the connection string fragment, key restrictions combination, and the key restriction behavior.

In the following example, KeyRestrictionBehavior.AllowOnly allows connection strings that use orcl as the Data Source with any User Id and Password combination but no other connection string keywords. Connection string keywords other than User Id and Password cause security exceptions.

```
orclPermission.Add("Data Source=orcl;", "User Id=;Password=;",  
    KeyRestrictionBehavior.AllowOnly);
```

In the next example, KeyRestrictionBehavior.PreventUsage restricts connection strings that use the keyword Pooling. Use of the Pooling keyword causes an exception.

```
orclPermission.Add("Data Source=orcl;", "Pooling=;",  
    KeyRestrictionBehavior.PreventUsage)
```

As a general rule, in an unrestricted environment, any connection string that is not allowed is restricted and throws a security exception.

If a connection string fragment contains key-value pairs for the password and proxy password attributes, then values for these attributes are ignored. However, the presence of the attributes themselves is still checked. This means that the connection is allowed only if the password and proxy attributes keywords are allowed in the connection string.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OraclePermission Class](#)
- [OraclePermission Members](#)

Copy

This method returns a copy of the current permission object.

Declaration

```
// C#  
public override IPermission Copy();
```


Return Value

A copy of the `OraclePermission` object.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OraclePermission Class](#)
- [OraclePermission Members](#)

IsSubsetOf

This method returns a boolean value that indicates whether or not the current permission is a subset of the target permission.

Declaration

```
// C#  
public override bool IsSubsetOf(IPermission target);
```

Parameters

- *target*
A permission that must be of type `OraclePermission`.

Return Value

A `bool` value that indicates whether or not the current permission is a subset of the target permission.

Exceptions

`ArgumentException` - The permission is not of the `OraclePermission` type.

Remarks

The `AllowBlankPassword` property is ignored when evaluating whether or not the current permission is a subset of the target permission.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OraclePermission Class](#)
- [OraclePermission Members](#)

OraclePermissionAttribute Class

An OraclePermissionAttribute object enables ODP.NET to enforce declarative security and helps ensure that a user has a security level adequate for accessing data.

Class Inheritance

System.Object
System.Attribute
System.Security.Permissions.SecurityAttribute
System.Security.Permissions.CodeAccessSecurityAttribute
System.Data.Common.DBDataPermissionAttribute
Oracle.DataAccess.Client.OraclePermissionAttribute

Declaration

```
// C#
[Serializable, AttributeUsage(AttributeTargets.Method |
AttributeTargets.Constructor | AttributeTargets.Class | AttributeTargets.Struct |
AttributeTargets.Assembly, AllowMultiple = true, Inherited = false)]
public sealed class OraclePermissionAttribute: DBDataPermissionAttribute
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OraclePermissionAttribute Members](#)
- [OraclePermissionAttribute Constructor](#)
- [OraclePermissionAttribute Static Methods](#)
- [OraclePermissionAttribute Public Properties](#)
- [OraclePermissionAttribute Public Methods](#)

OraclePermissionAttribute Members

OraclePermissionAttribute members are listed in the following tables.

OraclePermissionAttribute Constructor

The OraclePermissionAttribute constructor is listed in [Table 6-169](#).

Table 6-169 OraclePermission Constructor

Constructor	Description
OraclePermissionAttribute Constructor	Instantiates a new instance of the OraclePermissionAttribute class.

OraclePermissionAttribute Static Methods

The OraclePermissionAttribute static methods are listed in [Table 6-170](#).

Table 6-170 OraclePermissionAttribute Static Methods

Static Methods	Description
GetCustomAttribute	Inherited from System.Attribute (Overloaded)
GetCustomAttributes	Inherited from System.Attribute(Overloaded)
IsDefined	Inherited from System.Attribute(Overloaded)
ReferenceEquals	Inherited from System.Object

OraclePermissionAttribute Public Properties

The OraclePermissionAttribute public properties are listed in [Table 6-171](#).

Table 6-171 OraclePermissionAttribute Public Properties

Public Properties	Description
Action	Inherited from SecurityAttribute
AllowBlankPassword	Inherited from DBDataPermissionAttribute. OraclePermissionAttribute ignores this property. Any value set for this property, for an OraclePermissionAttribute object, is ignored.
ConnectionString	Inherited from DBDataPermissionAttribute
KeyRestrictionBehavior	Inherited from DBDataPermissionAttribute
KeyRestrictions	Inherited from DBDataPermissionAttribute
TypeId	Inherited from System.Attribute
Unrestricted	Inherited from SecurityAttribute

OraclePermissionAttribute Public Methods

The OraclePermissionAttribute public methods are listed in [Table 6-172](#).

Table 6-172 OraclePermissionAttribute Public Methods

Public Methods	Description
CreatePermission	Returns a new OraclePermissionAttribute object that is configured based on the attributes set
Equals	Inherited from System.Attribute
GetHashCode	Inherited from System.Attribute
GetType	Inherited from System.Attribute
IsDefaultAttribute	Inherited from System.Attribute
Match	Inherited from System.Attribute
ShouldSerializeConnectionString	Inherited from DBDataPermissionAttribute
ShouldSerializeKeyRestrictions	Inherited from DBDataPermissionAttribute
ToString	Inherited from System.Object

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OraclePermissionAttribute Class](#)

OraclePermissionAttribute Constructor

The OraclePermissionAttribute constructor instantiates new instances of the OraclePermissionAttribute class.

Declaration

```
// C#
public OraclePermissionAttribute (SecurityAction action);
```

Parameters

- *action*

A System.Security.Permissions.SecurityAction value representing an action that can be performed using declarative security.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OraclePermissionAttribute Class](#)
- [OraclePermissionAttribute Members](#)

OraclePermissionAttribute Static Methods

The OraclePermissionAttribute static methods are listed in [Table 6-173](#).

Table 6-173 OraclePermissionAttribute Static Methods

Static Methods	Description
GetCustomAttribute	Inherited from System.Attribute (Overloaded)
GetCustomAttributes	Inherited from System.Attribute(Overloaded)
IsDefined	Inherited from System.Attribute(Overloaded)
ReferenceEquals	Inherited from System.Object

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OraclePermissionAttribute Class](#)
- [OraclePermissionAttribute Members](#)

OraclePermissionAttribute Public Properties

The OraclePermissionAttribute public properties are listed in [Table 6-174](#).

Table 6-174 OraclePermissionAttribute Public Properties

Public Properties	Description
Action	Inherited from SecurityAttribute
AllowBlankPassword	Inherited from DBDataPermissionAttribute. OraclePermissionAttribute ignores this property. Any value set for this property, for an OraclePermissionAttribute object, is ignored.
ConnectionString	Inherited from DBDataPermissionAttribute
KeyRestrictionBehavior	Inherited from DBDataPermissionAttribute
KeyRestrictions	Inherited from DBDataPermissionAttribute
TypeId	Inherited from System.Attribute
Unrestricted	Inherited from SecurityAttribute

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OraclePermissionAttribute Class](#)
- [OraclePermissionAttribute Members](#)

OraclePermissionAttribute Public Methods

The OraclePermissionAttribute public methods are listed in [Table 6-175](#).

Table 6-175 OraclePermissionAttribute Public Methods

Public Methods	Description
CreatePermission	Returns a new OraclePermissionAttribute object that is configured based on the attributes set
Equals	Inherited from System.Attribute
GetHashCode	Inherited from System.Attribute
GetType	Inherited from System.Attribute
IsDefaultAttribute	Inherited from System.Attribute
Match	Inherited from System.Attribute
ShouldSerializeConnectionString	Inherited from DBDataPermissionAttribute
ShouldSerializeKeyRestrictions	Inherited from DBDataPermissionAttribute
ToString	Inherited from System.Object

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OraclePermissionAttribute Class](#)
- [OraclePermissionAttribute Members](#)

CreatePermission

This method returns a new OraclePermissionAttribute object that is configured based on the attributes set.

Declaration

```
// C#
public override IPPermission CreatePermission();
```

Return Value

An OraclePermission object.

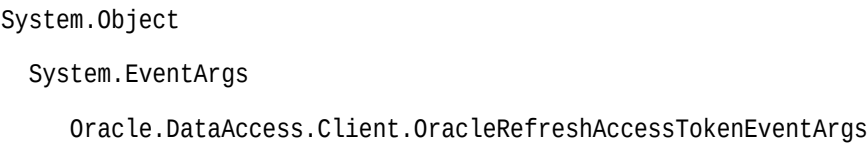
① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OraclePermissionAttribute Class](#)
- [OraclePermissionAttribute Members](#)

OracleRefreshAccessTokenEventArgs Class

The OracleRefreshAccessTokenEventArgs class provides a way for the application to return the refreshed signature token. When the token is about to expire, the callback registered using RefreshAccessToken event is triggered along with the OracleRefreshAccessTokenEventArgs object. The application is expected to provide the refreshed token with a new expiration within this object.

Class Inheritance



Declaration

```
// C#
public sealed class OracleRefreshAccessTokenEventArgs
```

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRefreshAccessTokenEventArgs Members](#)
- [OracleRefreshAccessTokenEventArgs Properties](#)

OracleRefreshAccessTokenEventArgs Members

OracleRefreshAccessTokenEventArgs members are listed in the following tables.

OracleRefreshAccessTokenEventArgs Properties

OracleRefreshAccessTokenEventArgs properties are listed in [Table 6-200](#).

Table 6-176 OracleRefreshAccessTokenEventArgs Properties

Property	Description
DbToken	Sets the new/refreshed signature token in this object
PrivateKey	Sets the refreshed private key in this object
Token	Sets the new or refreshed access token in this object.

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleRefreshAccessTokenEventArgs Class](#)

OracleRefreshAccessTokenEventArgs Properties

The OracleRefreshAccessTokenEventArgs properties are listed in [Table 6-187](#).

Table 6-177 OracleRefreshAccessTokenEventArgs Properties

Property	Description
DbToken	Sets the new/refreshed signature token in this object
PrivateKey	Sets the refreshed private key in this object
Token	Sets the new or refreshed access token in this object.

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleRefreshAccessTokenEventArgs Class](#)
- [OracleRefreshAccessTokenEventArgs Members](#)

DbToken

This property sets the new/refreshed signature token in this object.

Declaration

```
// C#  
public char[] DbToken { set; }
```


Description

The char[] used to set the value of this property will be cleared by ODP.NET. Applications should not depend on this char[] once this property is called.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleRefreshAccessTokenEventArgs Class](#)
- [OracleRefreshAccessTokenEventArgs Members](#)

PrivateKey

This property sets the refreshed private key in this object.

Declaration

```
// C#  
public char[] PrivateKey { set; }
```

Description

The char[] used to set the value of this property will be cleared by ODP.NET. Applications should not depend on this char[] once this property is called.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleRefreshAccessTokenEventArgs Class](#)
- [OracleRefreshAccessTokenEventArgs Members](#)

Token

This property sets the new or refreshed access token in this object.

Declaration

```
// C#  
public char[] Token { set; }
```

Description

This property can hold an access token.

The char[] used to set the value of this property will be cleared by ODP.NET for security reasons. The application should not depend on this char[] once this property is called.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRefreshAccessTokenEventArgs Class](#)
- [OracleRefreshAccessTokenEventArgs Members](#)

OracleRoleCollection Class

The OracleRoleCollection class represents a collection of roles that can be set on new connections or connections retrieved from a connection pool programmatically.

Class Inheritance

System.Object

Oracle.ManagedDataAccess.Client.OracleRoleCollection

Declaration

```
// C#  
public sealed class OracleRoleCollection
```

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

Thread Safety

Instance methods do not guarantee thread safety.

Remarks

Database role names are case sensitive. If a name without double-quotes is specified in the database, it automatically converts all its characters to upper case. If double quotes are specified, the upper and lower case letters are maintained in the database. Oracle does not allow starting a role name with single quote.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRoleCollection Members](#)
- [OracleRoleCollection Methods](#)
- [OracleRoleCollection Properties](#)

OracleRoleCollection Members

OracleRoleCollection members are listed in the following tables.

OracleRoleCollection Methods

The OracleSessionlessTransactionOptions methods are listed in [Table 6-178](#).

Table 6-178 OracleRoleCollection Methods

Methods	Description
Add	Adds a role to the role list (Overloaded)
Clear	Clears all the roles in the roles collection
Contains	Verifies whether a role is present in the roles collection or a role is enabled on a connection if the connection is open depending on the Oracle software versions
Remove	Removes a role from the roles list

OracleRoleCollection Properties

The OracleRoleCollection properties are listed in [Table 6-179](#).

Table 6-179 OracleRoleCollection Properties

Property	Description
Count	Specifies the number of roles in the OracleRoleCollection object or enabled on the current session
Item	Retrieves a role in the OracleRoleCollection object using an index value
UseAllExcept	Specifies enabling all non-password-based roles except those added in OracleRoleCollection

OracleRoleCollection Methods

OracleRoleCollection methods are listed in [Table 6-180](#).

Table 6-180 OracleRoleCollection Methods

Method	Description
Add	Adds a role to the role list (Overloaded)
Clear	Clears all the roles in the roles collection
Contains	Verifies whether a role is present in the roles collection or a role is enabled on a connection if the connection is open depending on the Oracle software versions
Remove	Removes a role from the roles list

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRoleCollection Class](#)
- [OracleRoleCollection Members](#)

Add

The Add method adds a new role to the role list.

Overloaded List:

- [Add\(string\)](#)
This method adds a role to the roles list and sets it on a on a newly created connection or a connection coming from connection pool.
- [Add\(string, OracleOpaqueString\)](#)
This method adds a role with a password to the roles list and sets it on a on a newly created connection or a connection coming from connection pool.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRoleCollection Class](#)
- [OracleRoleCollection Members](#)

Add(string)

This method adds a role to the roles list and sets it on a on a newly created connection or a connection coming from connection pool.

Declaration

```
// C#  
public void Add(string Role);
```

Parameters

Role - Role name that will be set on a connection.

Exceptions

`ArgumentNullException` - It is thrown when role name is provided as Null.

`ArgumentException` - It is thrown when role name is provided as empty string.

`ArgumentException` - It is thrown when case sensitive role name is not enclosed properly within double quotes.

`InvalidOperationException` - It is thrown when the connection is already open.

Remarks

The user must already be granted the database roles before adding those roles to the `OracleRoleCollection` object. Otherwise, an error occurs indicating the role is not granted or does not exist. The database roles not added to the ODP.NET collection are disabled for the session, except when no roles are added to the collection. In this case, the default roles are enabled in the database session.

To modify an existing connection's enabled roles, close the connection, modify the `OracleRoleCollection` instance associated with the connection instance, and reopen the connection.

To enable all roles granted, except roles with passwords, add ALL to the `OracleRoleCollection` instance. To disable all roles including default roles, add NONE to the `OracleRoleCollection` instance. ALL and NONE are reserved keywords for role names. When using ALL or NONE, make sure no other role names are added to the collection. Otherwise, it results in errors.

Once the session is created after inserting the ALL reserved keyword or when `OracleRoleCollection.UseAllExcept` property is set to true, ODP.NET does not modify the session's enabled roles if the database modifies the default roles until the `OracleRoleCollection` is itself modified. This holds true even for the connections which are checked into the pool.

To enable all previously granted user roles except for a particular collection, set `OracleRoleCollection.UseAllExcept` to true. It enables all the roles except those which are added in `OracleRoleCollection` object associated with `OracleConnection` object. If this property is set to True, then the password-based roles are not enabled on the connection. If `OracleRoleCollection.UseAllExcept` is true and nothing is added in the `OracleRoleCollection` object, then ODP.NET will not enable or disable any roles on the connection. The connection state with respect to roles enabled remains unchanged.

Note

For DRCP connections, because random sessions are checked out from server pool and it can have any set of roles enabled on the session, it is necessary to add the ALL keyword in the `OracleRoleCollection` object associated with the `OracleConnection` object, if you want to enable all the non-password based roles on the upcoming session.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRoleCollection Class](#)
- [OracleRoleCollection Members](#)

Add(string, OracleOpaqueString)

This method adds a role with a password to the roles list and sets it on a on a newly created connection or a connection coming from connection pool.

Declaration

```
// C#  
public void Add(string Role, OracleOpaqueString Password);
```

Parameters

- *Role* - Role name that will be set on a connection.
- *Password* - Password of Role

Exceptions

ArgumentNullException - It is thrown when role name is provided as Null.

ArgumentException - It is thrown when role name is provided as empty string.

ArgumentException - It is thrown when case sensitive role name is not enclosed properly within double quotes.

InvalidOperationException - It is thrown when the connection is already open.

Remarks

The user must already be granted the database roles before adding those roles to the OracleRoleCollection object. Otherwise, an error occurs indicating the role is not granted or does not exist. The database roles not added to the ODP.NET collection are disabled for the session, except when no roles are added to the collection. In this case, the non-password-based roles are enabled in the database session.

To modify an existing connection's enabled roles, close the connection, modify the OracleRoleCollection instance associated with the connection instance, and reopen the connection.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRoleCollection Class](#)
- [OracleRoleCollection Members](#)

Clear

This method clears all the roles in the roles collection.

Declaration

```
// C#  
public void clear();
```

Exceptions

InvalidOperationException - It is thrown when the connection is already open.

Remarks

This method only clears entries from the `OracleRoleCollection` object associated with the `OracleConnection` object. After clearing the roles and then opening a connection, ODP.NET does not enable nor disable any session roles due to the empty `OracleRoleCollection` object. For a session to have no roles enabled, add `NONE` to the `OracleRoleCollection` object.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRoleCollection Class](#)
- [OracleRoleCollection Members](#)

Contains

This method verifies whether a role is present in the roles collection or a role is enabled on a connection if the connection is open depending on the Oracle software versions.

Declaration

```
// C#  
public bool Contains(string Role)
```

Parameters

Role - Role name to check for presence in the roles collection.

Return Value

A Boolean value indicating if the role is present in the ODP.NET roles collection or that role is enabled in the current session. The exact behavior depends on the database version.

Exceptions

ArgumentNullException - It is thrown when role name is provided as Null.

ArgumentException - It is thrown when role name is provided as empty string.

ArgumentException - It is thrown when case sensitive role name is not enclosed properly within double quotes.

Remarks

For Oracle AI Database 26ai, 23.8 and earlier versions, this method only checks for roles specified in the `OracleConnection.Roles` property regardless whether the connection is open or closed.

For Oracle Database and ODP.NET versions 23.9 and later, this method checks if a given role is enabled on the current session, which requires the connection to be open. This method still returns true for roles not specified using `OracleRoleCollection.Add` but still are enabled on the current session. An example are default roles, which are enabled on the session as soon as it establishes.

For DRCP connections, until a session attaches to the `OracleConnection` object, this method searches the role name specified in the `OracleRoleCollection` object.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRoleCollection Class](#)
- [OracleRoleCollection Members](#)

Remove

This method removes a role from the roles list.

Declaration

```
// C#  
public bool Remove(string Role);
```

Parameters

Role - Role name to be removed from role list.

Return Value

A Boolean value indicating whether the specified role was removed from the list (true) or not (false).

Exceptions

`ArgumentNullException` - It is thrown when role name is provided as Null.

`ArgumentException` - It is thrown when role name is provided as empty string.

`ArgumentException` - It is thrown when case sensitive role name is not enclosed properly within double quotes.

`InvalidOperationException` - It is thrown when the connection is already open.

Remarks

A password-based role can also be removed from the collection by providing its name to the method.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRoleCollection Class](#)
- [OracleRoleCollection Members](#)

OracleRoleCollection Properties

OracleRoleCollection properties are listed in [Table 6-181](#).

Table 6-181 OracleRoleCollection Properties

Property	Description
Count	Specifies the number of roles in the OracleRoleCollection object or enabled on the current session
Item	Retrieves a role in the OracleRoleCollection object using an index value
UseAllExcept	Specifies enabling all non-password-based roles except those added in OracleRoleCollection

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRoleCollection Class](#)
- [OracleRoleCollection Members](#)

Count

This property specifies the number of roles in the OracleRoleCollection object or enabled on the current session.

Declaration

```
// C#
public int Count{get;}
```

Property Value

Number of roles in this OracleRoleCollection object or enabled on current session.

Remarks

For Oracle AI Database 26ai, 23.8 and earlier versions, this property only returns the total number of roles specified in the OracleConnection.Roles property regardless whether the connection is open or closed.

For Oracle Database and ODP.NET versions 23.9 and later, this property returns the number of roles enabled on a connection when the connection is open.

When a reserved keyword for roles, such as ALL or NONE, is part of the `OracleRoleCollection` object, the property value is the total items added to the object when the connection is closed. When the connection is open, this property returns the total number of roles enabled on the session.

For DRCP connections, until a session attaches to the `OracleConnection` object, this property returns the total roles specified in `OracleRoleCollection` object.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRoleCollection Class](#)
- [OracleRoleCollection Members](#)

Item

This property retrieves a role in the `OracleRoleCollection` object using an index value.

Declaration

```
// C#  
public string Item[int index] {get;}
```

Parameters

- *index* - An integer that represents the index value for role name retrieval.

Property Value

A string for the role name

Exceptions

`ArgumentOutOfRangeException` - Index is outside the range of valid indexes.

Remarks

Roles are stored in alphabetical order by role name. To fetch the role name, the index value should be less than the `OracleRoleCollection.Count` value. Otherwise, it results in an `ArgumentOutOfRangeException`.

For Oracle AI Database 26ai, 23.8 and earlier versions, this property only returns roles specified in `OracleConnection.Roles` property regardless whether the connection is open or closed.

For Oracle AI Database and ODP.NET versions 23.9 and later, this property returns the default session roles names enabled on the session when the connection is open.

When a reserved keyword for roles, such as ALL or NONE, is part of the `OracleRoleCollection` object, this property returns the items added in the roles collection when the connection is closed. When the connection is open, this property returns the session-enabled role names. Similarly, when `OracleRoleCollection.UseAllExcept` property is set to true and connection is

closed, this property returns the `OracleRoleCollection` item names. When the connection is open, this property returns session-enabled role names.

For DRCP connections, this property returns the role names specified in the `OracleRoleCollection` object until the connection opens and a session is attached to it.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRoleCollection Class](#)
- [OracleRoleCollection Members](#)

UseAllExcept

This property specifies enabling all non-password-based roles except those added in `OracleRoleCollection`.

Declaration

```
// C#  
public bool UseAllExcept{get;set;}
```

Property Value

A Boolean specifying whether to enable all non-password-based roles except those added in `OracleRoleCollection`.

Remarks

The roles in `OracleRoleCollection` must be directly granted to the database user. They cannot be the roles granted to that user via other roles.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRoleCollection Class](#)
- [OracleRoleCollection Members](#)

OracleRowUpdatedEventArgs Class

The `OracleRowUpdatedEventArgs` class provides event data for the `OracleDataAdapter.RowUpdated` event.

Class Inheritance

`System.Object`

`System.EventArgs`

`System.RowUpdatedEventArgs`

System.OracleRowUpdatedEventArgs

Declaration

```
// C#
public sealed class OracleRowUpdatedEventArgs : RowUpdatedEventArgs
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

The example for the RowUpdated event shows how to use OracleRowUpdatedEventArgs. See RowUpdated event "[Example](#)".

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleRowUpdatedEventArgs Members](#)
- [OracleRowUpdatedEventArgs Constructor](#)
- [OracleRowUpdatedEventArgs Static Methods](#)
- [OracleRowUpdatedEventArgs Properties](#)
- [OracleRowUpdatedEventArgs Public Methods](#)
- [OracleDataAdapter Class](#)

OracleRowUpdatedEventArgs Members

OracleRowUpdatedEventArgs members are listed in the following tables.

OracleRowUpdatedEventArgs Constructors

OracleRowUpdatedEventArgs constructors are listed in [Table 6-182](#).

Table 6-182 OracleRowUpdatedEventArgs Constructors

Constructor	Description
OracleRowUpdatedEventArgs Constructor	Instantiates a new instance of OracleRowUpdatedEventArgs class

OracleRowUpdatedEventArgs Static Methods

The OracleRowUpdatedEventArgs static method is listed in [Table 6-183](#).

Table 6-183 OracleRowUpdatedEventArgs Static Method

Method	Description
Equals	Inherited from System.Object (Overloaded)

OracleRowUpdatedEventArgs Properties

The OracleRowUpdatedEventArgs properties are listed in [Table 6-184](#).

Table 6-184 OracleRowUpdatedEventArgs Properties

Property	Description
Command	Specifies the OracleCommand that is used when OracleDataAdapter.Update() is called
Errors	Inherited from System.Data.Common.RowUpdatedEventArgs
RecordsAffected	Inherited from System.Data.Common.RowUpdatedEventArgs
Row	Inherited from System.Data.Common.RowUpdatedEventArgs
StatementType	Inherited from System.Data.Common.RowUpdatedEventArgs
Status	Inherited from System.Data.Common.RowUpdatedEventArgs
TableMapping	Inherited from System.Data.Common.RowUpdatedEventArgs

OracleRowUpdatedEventArgs Public Methods

The OracleRowUpdatedEventArgs properties are listed in [Table 6-185](#).

Table 6-185 OracleRowUpdatedEventArgs Public Methods

Public Method	Description
Equals	Inherited from System.Object (Overloaded)
GetHashCode	Inherited from System.Object
GetType	Inherited from System.Object
ToString	Inherited from System.Object

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRowUpdatedEventArgs Class](#)

OracleRowUpdatedEventArgs Constructor

The `OracleRowUpdatedEventArgs` constructor creates a new `OracleRowUpdatedEventArgs` instance.

Declaration

```
// C#
public OracleRowUpdatedEventArgs(DataRow row, IDbCommand command,
    StatementType statementType, DataTableMapping tableMapping);
```

Parameters

- *row*
The `DataRow` sent for Update.
- *command*
The `IDbCommand` executed during the Update.
- *statementType*
The `StatementType` Enumeration value indicating the type of SQL statement executed.
- *tableMapping*
The `DataTableMapping` used for the Update.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRowUpdatedEventArgs Class](#)
- [OracleRowUpdatedEventArgs Members](#)

OracleRowUpdatedEventArgs Static Methods

The `OracleRowUpdatedEventArgs` static method is listed in [Table 6-186](#).

Table 6-186 OracleRowUpdatedEventArgs Static Method

Method	Description
<code>Equals</code>	Inherited from <code>System.Object</code> (Overloaded)

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRowUpdatedEventArgs Class](#)
- [OracleRowUpdatedEventArgs Members](#)

OracleRowUpdatedEventArgs Properties

The OracleRowUpdatedEventArgs properties are listed in [Table 6-187](#).

Table 6-187 OracleRowUpdatedEventArgs Properties

Property	Description
Command	Specifies the OracleCommand that is used when OracleDataAdapter.Update() is called
Errors	Inherited from System.Data.Common.RowUpdatedEventArgs
RecordsAffected	Inherited from System.Data.Common.RowUpdatedEventArgs
Row	Inherited from System.Data.Common.RowUpdatedEventArgs
StatementType	Inherited from System.Data.Common.RowUpdatedEventArgs
Status	Inherited from System.Data.Common.RowUpdatedEventArgs
TableMapping	Inherited from System.Data.Common.RowUpdatedEventArgs

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRowUpdatedEventArgs Class](#)
- [OracleRowUpdatedEventArgs Members](#)

Command

This property specifies the OracleCommand that is used when OracleDataAdapter.Update() is called.

Declaration

```
// C#
public new OracleCommand Command {get;}
```

Property Value

The OracleCommand executed when Update is called.

❗ See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRowUpdatedEventArgs Class](#)
- [OracleRowUpdatedEventArgs Members](#)

OracleRowUpdatedEventArgs Public Methods

The OracleRowUpdatedEventArgs properties are listed in [Table 6-188](#).

Table 6-188 OracleRowUpdatedEventArgs Public Methods

Public Method	Description
Equals	Inherited from System.Object (Overloaded)
GetHashCode	Inherited from System.Object
GetType	Inherited from System.Object
ToString	Inherited from System.Object

❗ See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRowUpdatedEventArgs Class](#)
- [OracleRowUpdatedEventArgs Members](#)

OracleRowUpdatedEventHandler Delegate

The OracleRowUpdatedEventHandler delegate represents the signature of the method that handles the OracleDataAdapter.RowUpdated event.

Declaration

```
// C#
public delegate void OracleRowUpdatedEventHandler(object sender,
    OracleRowUpdatedEventArgs eventArgs);
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Parameters

- *sender*
The source of the event.
- *eventArgs*
The OracleRowUpdatedEventArgs object that contains the event data.

Remarks

Event callbacks can be registered through this event delegate for applications that wish to be notified after a row is updated.

In the .NET framework, the convention of an event delegate requires two parameters: the object that raises the event and the event data.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- "[RowUpdated](#)"

OracleRowUpdatingEventArgs Class

The OracleRowUpdatingEventArgs class provides event data for the OracleDataAdapter.RowUpdating event.

Class Inheritance

System.Object

System.EventArgs

System.RowUpdatingEventArgs

System.OracleRowUpdatingEventArgs

Declaration

```
// C#
public sealed class OracleRowUpdatingEventArgs : RowUpdatingEventArgs
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

The example for the RowUpdated event shows how to use OracleRowUpdatingEventArgs. See RowUpdated event "[Example](#)".

 See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleRowUpdatingEventArgs Members](#)
- [OracleRowUpdatingEventArgs Constructor](#)
- [OracleRowUpdatingEventArgs Static Methods](#)
- [OracleRowUpdatingEventArgs Properties](#)
- [OracleRowUpdatingEventArgs Public Methods](#)
- "[OracleDataAdapter Class](#) "

OracleRowUpdatingEventArgs Members

OracleRowUpdatingEventArgs members are listed in the following tables.

OracleRowUpdatingEventArgs Constructors

OracleRowUpdatingEventArgs constructors are listed in [Table 6-189](#).

Table 6-189 OracleRowUpdatingEventArgs Constructors

Constructor	Description
OracleRowUpdatingEventArgs Constructor	Instantiates a new instance of OracleRowUpdatingEventArgs class (Overloaded)

OracleRowUpdatingEventArgs Static Methods

The OracleRowUpdatingEventArgs static methods are listed in [Table 6-190](#).

Table 6-190 OracleRowUpdatingEventArgs Static Methods

Method	Description
Equals	Inherited from System.Object (Overloaded)

OracleRowUpdatingEventArgs Properties

The OracleRowUpdatingEventArgs properties are listed in [Table 6-191](#).

Table 6-191 OracleRowUpdatingEventArgs Properties

Property	Description
Command	Specifies the OracleCommand that is used when the OracleDataAdapter.Update() is called
Errors	Inherited from System.Data.Common.RowUpdatingEventArgs
Row	Inherited from System.Data.Common.RowUpdatingEventArgs
StatementType	Inherited from System.Data.Common.RowUpdatingEventArgs
Status	Inherited from System.Data.Common.RowUpdatingEventArgs
TableMapping	Inherited from System.Data.Common.RowUpdatingEventArgs

OracleRowUpdatingEventArgs Public Methods

The OracleRowUpdatingEventArgs public methods are listed in [Table 6-192](#).

Table 6-192 OracleRowUpdatingEventArgs Public Methods

Public Method	Description
Equals	Inherited from System.Object (Overloaded)
GetHashCode	Inherited from System.Object
GetType	Inherited from System.Object
ToString	Inherited from System.Object

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRowUpdatingEventArgs Class](#)

OracleRowUpdatingEventArgs Constructor

The OracleRowUpdatingEventArgs constructor creates a new instance of the OracleRowUpdatingEventArgs class using the supplied data row, IDbCommand, type of SQL statement, and table mapping.

Declaration

```
// C#
public OracleRowUpdatingEventArgs(DataRow row, IDbCommand command,
    StatementType statementType, DataTableMapping tableMapping);
```

Parameters

- row*
The DataRow sent for Update.
- command*
The IDbCommand executed during the Update.
- statementType*
The StatementType enumeration value indicating the type of SQL statement executed.
- tableMapping*
The DataTableMapping used for the Update.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRowUpdatingEventArgs Class](#)
- [OracleRowUpdatingEventArgs Members](#)

OracleRowUpdatingEventArgs Static Methods

The OracleRowUpdatingEventArgs static method is listed in [Table 6-193](#).

Table 6-193 OracleRowUpdatingEventArgs Static Method

Method	Description
Equals	Inherited from System.Object (Overloaded)

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRowUpdatingEventArgs Class](#)
- [OracleRowUpdatingEventArgs Members](#)

OracleRowUpdatingEventArgs Properties

The OracleRowUpdatingEventArgs properties are listed in [Table 6-194](#).

Table 6-194 OracleRowUpdatingEventArgs Properties

Property	Description
Command	Specifies the OracleCommand that is used when the OracleDataAdapter.Update() is called
Errors	Inherited from System.Data.Common.RowUpdatingEventArgs
Row	Inherited from System.Data.Common.RowUpdatingEventArgs
StatementType	Inherited from System.Data.Common.RowUpdatingEventArgs
Status	Inherited from System.Data.Common.RowUpdatingEventArgs
TableMapping	Inherited from System.Data.Common.RowUpdatingEventArgs

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRowUpdatingEventArgs Class](#)
- [OracleRowUpdatingEventArgs Members](#)

Command

This property specifies the OracleCommand that is used when the OracleDataAdapter.Update() is called.

Declaration

```
// C#  
public new OracleCommand Command {get; set;}
```

Property Value

The OracleCommand executed when Update is called.

❗ See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRowUpdatingEventArgs Class](#)
- [OracleRowUpdatingEventArgs Members](#)

OracleRowUpdatingEventArgs Public Methods

The OracleRowUpdatingEventArgs public methods are listed in [Table 6-195](#).

Table 6-195 OracleRowUpdatingEventArgs Public Methods

Public Method	Description
Equals	Inherited from System.Object (Overloaded)
GetHashCode	Inherited from System.Object
GetType	Inherited from System.Object
ToString	Inherited from System.Object

❗ See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRowUpdatingEventArgs Class](#)
- [OracleRowUpdatingEventArgs Members](#)

OracleRowUpdatingEventHandler Delegate

The OracleRowUpdatingEventHandler delegate represents the signature of the method that handles the OracleDataAdapter.RowUpdating event.

Declaration

```
// C#
public delegate void OracleRowUpdatingEventHandler (object sender,
    OracleRowUpdatingEventArgs eventArgs);
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Parameters

- *sender*
The source of the event.
- *eventArgs*
The OracleRowUpdatingEventArgs object that contains the event data.

Remarks

Event callbacks can be registered through this event delegate for applications that wish to be notified after a row is updated.

In the .NET framework, the convention of an event delegate requires two parameters: the object that raises the event and the event data.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- "[RowUpdating](#)"

OracleSessionlessTransactionOptions Class

The OracleSessionlessTransactionOption class specifies the options to operate on a transaction that can span across multiple sessions. This transaction type is called an Oracle sessionless transaction.

Class Inheritance

System.Object

Oracle.ManagedDataAccess.Client.OracleSessionlessTransactionOption

Declaration

```
// C#
public class OracleSessionlessTransactionOptions
```

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSessionlessTransactionOptions Members](#)
- [OracleSessionlessTransactionOptions Constructors](#)
- [OracleSessionlessTransactionOptions Properties](#)

OracleSessionlessTransactionOptions Members

OracleSessionlessTransactionOptions members are listed in the following tables.

OracleSessionlessTransactionOptions Constructors

The OracleSessionlessTransactionOptions constructors are listed in [Table 6-196](#).

Table 6-196 OracleSessionlessTransactionOptions Constructors

Constructors	Description
OracleSessionlessTransactionOptions Constructors	Instantiate new instances of an OracleSessionlessTransactionOptions class

OracleSessionlessTransactionOptions Properties

The OracleSessionlessTransactionOptions properties are listed in [Table 6-197](#).

Table 6-197 OracleSessionlessTransactionOptions Properties

Property	Description
ResumeBehavior	Specifies sessionless transaction resume behavior in <code>OracleConnection.ResumeSessionlessTransaction()</code>
StartBehavior	Specifies sessionless transaction start behavior in <code>OracleConnection.BeginSessionlessTransaction()</code>
SuspendAfterExecution	Specifies whether the active sessionless transaction should be suspended after an <code>OracleCommand</code> SQL execution

Table 6-197 (Cont.) OracleSessionlessTransactionOptions Properties

Property	Description
Timeout	Specifies the time, in seconds, to wait for a transaction to become active after it is detached or suspended

OracleSessionlessTransactionOptions Constructors

OracleSessionlessTransactionOptions constructors instantiate new instances of an OracleSessionlessTransactionOptions class.

Overload List:

- [OracleSessionlessTransactionOptions\(\)](#)
This constructor instantiates a new instance of the OracleSessionlessTransactionOptions class using default property values.
- [OracleSessionlessTransactionOptions\(int, OracleSessionlessTransactionStartBehavior, OracleSessionlessTransactionStartBehavior, bool\)](#)
This constructor instantiates a new instance of the OracleSessionlessTransactionOptions class with the provided transaction options.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSessionlessTransactionOptions Class](#)
- [OracleSessionlessTransactionOptions Members](#)

OracleSessionlessTransactionOptions()

This constructor instantiates a new instance of the OracleSessionlessTransactionOptions class using default property values.

Declaration

```
// C#  
public OracleSessionlessTransactionOptions();
```

Remarks

OracleSessionlessTransactionOptions properties have the following default values:

- Timeout = 60 seconds
- StartBehavior = OracleSessionlessTransactionStartBehavior.StartBeforeExecute
- ResumeBehavior = OracleSessionlessTransactionStartBehavior.StartBeforeExecute
- SuspendSessionlessTransactionAfterExecution = false

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSessionlessTransactionOptions Class](#)
- [OracleSessionlessTransactionOptions Members](#)

OracleSessionlessTransactionOptions(int, OracleSessionlessTransactionStartBehavior, OracleSessionlessTransactionStartBehavior, bool)

This constructor instantiates a new instance of the `OracleSessionlessTransactionOptions` class with the provided transaction options.

Declaration

```
// C#
public OracleSessionlessTransactionOptions(int timeout,
OracleSessionlessTransactionStartBehavior startBehavior,
OracleSessionlessTransactionStartBehavior resumeBehavior,
bool suspendAfterExecution);
```

Parameters

- *timeout* : time, in seconds, to wait for a transaction to become active after it is detached or suspended. If a non-active sessionless transaction is not resumed, committed, or rolled back within the timeout period, it is aborted. The timeout value must be set greater than 0.
- *startBehavior* : determines if the sessionless transaction should start immediately in `OracleConnection.BeginSessionlessTransaction()` or upon an `OracleCommand` SQL execution.
- *resumeBehavior* : determines if the sessionless transaction should resume immediately in `OracleConnection.ResumeSessionlessTransaction()` or upon an `OracleCommand` SQL execution.
- *suspendAfterExecution* : determines if the active sessionless transaction should be suspended after an `OracleCommand` SQL execution.

Exceptions

`ArgumentOutOfRangeException` - specified timeout is equal or smaller than zero.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSessionlessTransactionOptions Class](#)
- [OracleSessionlessTransactionOptions Members](#)

OracleSessionlessTransactionOptions Properties

OracleSessionlessTransactionOptions properties are listed in [Table 6-198](#).

Table 6-198 OracleSessionlessTransactionOptions Properties

Property	Description
ResumeBehavior	Specifies sessionless transaction resume behavior in <code>OracleConnection.ResumeSessionlessTransaction()</code>
StartBehavior	Specifies sessionless transaction start behavior in <code>OracleConnection.BeginSessionlessTransaction()</code>
SuspendAfterExecution	Specifies whether the active sessionless transaction should be suspended after an <code>OracleCommand</code> SQL execution
Timeout	Specifies the time, in seconds, to wait for a transaction to become active after it is detached or suspended

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSessionlessTransactionOptions Class](#)
- [OracleSessionlessTransactionOptions Members](#)

ResumeBehavior

This property specifies sessionless transaction resume behavior in `OracleConnection.ResumeSessionlessTransaction()`.

Declaration

```
// C#  
public OracleSessionlessTransactionStartBehavior ResumeBehavior { get; set; }
```

Property Value

The `OracleSessionlessTransactionStartBehavior` enum specifies the behavior to resume a sessionless transaction.

Remarks

When `ResumeBehavior` is set to `OracleSessionlessTransactionStartBehavior.StartBeforeExecution`, the connection is associated with the sessionless transaction in `OracleConnection.ResumeSessionlessTransaction()`. However, the transaction will only resume upon the first `OracleCommand` SQL execution.

When `ResumeBehavior` is set to `OracleSessionlessTransactionStartBehavior.StartImmediate`, the sessionless transaction resumes immediately when `OracleConnection.ResumeSessionlessTransaction()` is invoked.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSessionlessTransactionOptions Class](#)
- [OracleSessionlessTransactionOptions Members](#)

StartBehavior

This property specifies sessionless transaction start behavior in `OracleConnection.BeginSessionlessTransaction()`.

Declaration

```
// C#  
public OracleSessionlessTransactionStartBehavior StartBehavior { get; set; }
```

Property Value

The `OracleSessionlessTransactionStartBehavior` enum that specifies sessionless transaction start behavior.

Remarks

When `StartBehavior` is set to `OracleSessionlessTransactionStartBehavior.StartBeforeExecution`, the connection is associated with the sessionless transaction in `OracleConnection.BeginSessionlessTransaction()`. However, the transaction will only start upon the first `OracleCommand` SQL execution.

When `StartBehavior` is set to `OracleSessionlessTransactionStartBehavior.StartImmediate`, the sessionless transaction starts immediately when `OracleConnection.BeginSessionlessTransaction()` is invoked.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSessionlessTransactionOptions Class](#)
- [OracleSessionlessTransactionOptions Members](#)

SuspendAfterExecution

This property specifies whether the active sessionless transaction should be suspended after an `OracleCommand` SQL execution.

Declaration

```
// C#  
public bool SuspendAfterExecution { get; set; }
```

Property Value

The boolean indicates if the sessionless transaction should be suspended after the OracleCommand SQL execution.

Remarks

When SuspendAfterExecution is set to true, then the active sessionless transaction is suspended upon the first OracleCommand SQL execution.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSessionlessTransactionOptions Class](#)
- [OracleSessionlessTransactionOptions Members](#)

Timeout

This property specifies the time, in seconds, to wait for a transaction to become active after it is detached or suspended.

Declaration

```
// C#  
public int Timeout { get; set; }
```

Property Value

The integer specifies the sessionless transaction timeout.

Exceptions

ArgumentOutOfRangeException - specified timeout is equal or smaller than zero.

Remarks

If a non-active sessionless transaction is not resumed, committed, or rolled back within the timeout period, then it is aborted. The timeout value must be set greater than 0.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSessionlessTransactionOptions Class](#)
- [OracleSessionlessTransactionOptions Members](#)

OracleShardingKey Class

An OracleShardingKey object can represent either a sharding key or a super sharding key.

Class Inheritance

System.Object

Oracle.DataAccess.Client.OracleShardingKey

Declaration

```
// C#
public class OracleShardingKey : IDisposable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Example

```
// C#

using System;
using Oracle.DataAccess.Client;

class Sharding
{
    static void Main()
    {
        OracleConnection con = new OracleConnection("user id=hr;password=hr;DataSource=orcl;");
        //Setting a shard key
        OracleShardingKey shardingKey = new OracleShardingKey(OracleDbType.Int32, 123);
        //Setting a second shard key value for a composite key
        shardingKey.SetShardingKey(OracleDbType.Varchar2, "gold");
        //Creating and setting the super shard key
        OracleShardingKey superShardingKey = new OracleShardingKey();
        superShardingKey.SetShardingKey(OracleDbType.Int32, 1000);

        //Setting super sharding key and sharding key on the connection
        con.SetShardingKey(shardingKey, superShardingKey);
        con.Open();

        //perform SQL query
    }
}
```

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleShardingKey Members](#)
- [OracleShardingKey Constructors](#)
- [OracleShardingKey Instance Methods](#)

OracleShardingKey Members

OracleShardingKey members are listed in the following tables.

OracleShardingKey Constructors

OracleShardingKey constructors are listed in [Table 6-199](#).

Table 6-199 OracleShardingKey Constructors

Constructor	Description
OracleShardingKey Constructors	Instantiates a new instance of OracleShardingKey class (Overloaded)

OracleShardingKey Instance Methods

OracleShardingKey instance methods are listed in [Table 6-200](#).

Table 6-200 OracleShardingKey Instance Methods

Method	Description
SetShardingKey(OracleDbType, object)	Enables applications to set a key within the OracleShardingKey object
Dispose	Enables applications to explicitly dispose the OracleShardingKey object

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleShardingKey Class](#)

OracleShardingKey Constructors

OracleShardingKey constructors instantiate new instances of the OracleShardingKey class.

Overload List:

- [OracleShardingKey\(\)](#)

This constructor instantiates a new instance of `OracleShardingKey` class.

- [OracleShardingKey\(OracleDbType, object\)](#)

This constructor instantiates a new instance of the `OracleShardingKey` class using the supplied data type and key.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleShardingKey Class](#)
- [OracleShardingKey Members](#)

OracleShardingKey()

This constructor enables applications to construct the `OracleShardingKey` object.

Declaration

```
// C#  
public OracleShardingKey();
```

Exceptions

None

Remarks

This constructs an `OracleShardingKey` without any keys set.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleShardingKey Class](#)
- [OracleShardingKey Members](#)

OracleShardingKey(OracleDbType, object)

This constructor enables applications to construct the `OracleShardingKey` object with the supplied key.

Declaration

```
// C#  
public OracleShardingKey(OracleDbType type, object key);
```

Exceptions

`InvalidArgumentException` – The supplied argument is invalid

Remarks

This constructs an OracleShardingKey with the supplied key set.

Acceptable OracleDbType enumeration values are Byte, Decimal, Double, Int16, Int32, Int64, Single, Varchar2, String, Date, TimeStamp, and Raw.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleShardingKey Class](#)
- [OracleShardingKey Members](#)

OracleShardingKey Instance Methods

OracleShardingKey instance methods are listed in [Table 6-201](#).

Table 6-201 OracleShardingKey Instance Methods

Instance Method	Description
SetShardingKey(OracleDbType type, object)	Enables applications to set a key within the OracleShardingKey object
Dispose	Enables applications to explicitly dispose the OracleShardingKey object

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleShardingKey Class](#)
- [OracleShardingKey Members](#)

SetShardingKey(OracleDbType, object)

This instance method enables applications to set a key within the OracleShardingKey object.

Declaration

```
// C#  
public void SetShardingKey(OracleDbType type, object key);
```

Exceptions

InvalidArgumentException – The supplied argument is invalid

Remarks

This method sets a key within the OracleShardingKey object.

Acceptable OracleDbType enumeration values are Byte, Decimal, Double, Int16, Int32, Int64, Single, Varchar2, String, Date, TimeStamp, and Raw.

This can be called multiple times to construct a composite key.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleShardingKey Class](#)
- [OracleShardingKey Members](#)

Dispose

This instance method enables applications to explicitly dispose the OracleShardingKey object.

Declaration

```
// C#  
public void Dispose();
```

Exceptions

None

Remarks

This method disposes the OracleShardingKey object.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleShardingKey Class](#)
- [OracleShardingKey Members](#)

OracleSparseVector Class

An OracleSparseVector object represents a sparse Oracle VECTOR.

Class Inheritance

System.Object

Oracle.DataAccess.Types.OracleVector

Oracle.DataAccess.Types.OracleSparseVector

Declaration

```
// C#  
public class OracleSparseVector : OracleVector
```

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [Artificial Intelligence Vectors and Semantic Search](#)
- [OracleVector Class](#)
- [OracleSparseVector Members](#)
- [OracleSparseVector Constructors](#)
- [OracleSparseVector Properties](#)
- [OracleSparseVector Instance Methods](#)

OracleSparseVector Members

OracleSparseVector members are listed in the following tables.

OracleSparseVector Constructors

The OracleSparseVector constructors is listed in [Table 6-202](#).

Table 6-202 OracleSparseVector Constructors

Constructor	Description
OracleSparseVector Constructors	OracleSparseVector constructors instantiate new instances of OracleSparseVector class. (Overloaded)

OracleSparseVector Properties

OracleSparseVector properties are listed in [Table 6-203](#).

Table 6-203 OracleSparseVector Properties

Property	Description
Dimensions	Returns the number of dimensions of the OracleSparseVector
VectorType	Returns the vector type
ProviderType	Returns the OracleDbType value of the OracleSparseVector provider type
Value	Returns a .NET numeric array representing the sparse vector values at the non-zero indices
Indices	Returns a long[] of the zero-based sparse vector's non-zero indices
Item	Obtains the element in an OracleSparseVector at the specified index
IsNull	Indicates whether the current instance has a null value

OracleSparseVector Static Field

OracleSparseVector static fields are listed in [Table 6-204](#).

Table 6-204 OracleSparseVector Static Field

Static Field	Description
Null	static readonly OracleSparseVector Null represents a null value that can be assigned to an OracleSparseVector instance.

OracleSparseVector Instance Methods

OracleSparseVector Instance methods are listed in [Table 6-205](#).

Table 6-205 OracleSparseVector Instance Methods

Instance Method	Description
ToInt16Array()	Returns a short[] value of the OracleSparseVector instance
ToFloatArray()	Returns a float[] value of the OracleSparseVector instance
ToDoubleArray()	Returns a double[] value of this OracleSparseVector instance
ToNonZeroInt16Array()	Returns a short[] of the sparse vector values at the non-zero indices
ToNonZeroFloatArray()	Returns a float[] of the sparse vector values at the non-zero indices
ToNonZeroDoubleArray()	Returns a double[] of the sparse vector values at the non-zero indices
ToString()	Returns the OracleSparseVector string representation
Dispose()	Releases resources allocated by this object

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSparseVector Class](#)

OracleSparseVector Constructors

OracleSparseVector constructors instantiate new instances of OracleSparseVector class.

Overload List:

- [OracleSparseVector\(sparseVectorString\)](#)
This constructor creates an instance of the OracleSparseVector class with the string representation of a sparse vector.
- [OracleSparseVector\(sparseVectorString, providerType\)](#)
This constructor creates an instance of the OracleSparseVector class with the string representation of a sparse vector.
- [OracleSparseVector\(dimensions, indices\)](#)
This constructor creates an instance of the OracleSparseVector class with a specified number of dimensions and the non-zero indices stored in a long[].
- [OracleSparseVector\(dimensions, indices, sparseValue\)](#)
This constructor creates an instance of the OracleSparseVector class with a specified number of dimensions and non-zero vector values stored in a short[].
- [OracleSparseVector\(dimensions, indices, sparseValue\)](#)
This constructor creates an instance of the OracleSparseVector class with a specified number of dimensions and non-zero vector values stored in a float[].
- [OracleSparseVector\(dimensions, indices, sparseValue\)](#)
This constructor creates an instance of the OracleSparseVector class with a specified number of dimensions and non-zero vector values stored in a double[].

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSparseVector Class](#)
- [OracleSparseVector Members](#)

OracleSparseVector(sparseVectorString)

This constructor creates an instance of the OracleSparseVector class with the string representation of a sparse vector.

Declaration

```
// C#  
public OracleSparseVector(string sparseVectorString);
```

Parameters

sparseVectorString

The string representation of a sparse vector

Exceptions

ArgumentException - The parameter is an invalid string representation of an OracleSparseVector.

ArgumentNullException - The parameter is null.

Remarks

The ProviderType is OracleDbType.Vector_Float32 unless it is in binary sparse vector format. If that is the case, the ProviderType is OracleDbType.Vector_Binary.

If the vector parameter has a length of zero, then the constructed instance will have null value and IsNull will be true.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSparseVector Class](#)
- [OracleSparseVector Members](#)

OracleSparseVector(sparseVectorString, providerType)

This constructor creates an instance of the OracleSparseVector class with the string representation of a sparse vector.

Declaration

```
// C#  
public OracleSparseVector(string sparseVectorString, OracleDbType providerType);
```

Parameters

- sparseVectorString
The string representation of a sparse vector
- providerType
The OracleDbType of the resulting OracleSparseVector.

Exceptions

ArgumentException - The parameter is an invalid string or binary representation of an OracleSparseVector or the providerType is not an OracleDbType.Vector*.

ArgumentNullException - The parameter is null.

Remarks

The allowed providerType values are:

- OracleDbType.Vector
- OracleDbType.Vector_Binary
- OracleDbType.Vector_Int8
- OracleDbType.Vector_Float32
- OracleDbType.Vector_Float64

If the vector parameter has a length of zero, then the constructed instance will have null value and IsNull will be true.

When providerType is OracleDbType.Vector and if sparseVectorString is in binary sparse vector format, then the OracleSparseVector ProviderType becomes OracleDbType.Vector_Binary; otherwise, it becomes OracleDbType.Vector_Float32 for other vector formats.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSparseVector Class](#)
- [OracleSparseVector Members](#)

OracleSparseVector(dimensions, indices)

This constructor creates an instance of the OracleSparseVector class with a specified number of dimensions and the non-zero indices stored in a long[].

Declaration

```
// C#  
public OracleSparseVector(long dimensions, long[] indices);
```

Parameters

- dimensions
The number of dimensions of the sparse vector
- indices
The long[] for the zero-based sparse vector's non-zero indices

Exceptions

ArgumentOutOfRangeException - index is less than 0, or an element in indices is less than 0 or greater than or equal to dimensions.

ArgumentNullException - The indices parameter is null.

ArgumentException - The dimensions are not a multiple of 8 or are less than the length of indices.

Remarks

The `ProviderType` is `OracleDbType.Vector_Binary`.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSparseVector Class](#)
- [OracleSparseVector Members](#)

OracleSparseVector(dimensions, indices, sparseValue)

This constructor creates an instance of the `OracleSparseVector` class with a specified number of dimensions and non-zero vector values stored in a `short[]`.

Declaration

```
// C#  
public OracleSparseVector(long dimensions, long[] indices, short[] sparseValue);
```

Parameters

- `dimensions`
The number of sparse vector dimensions.
- `indices`
The `long[]` for the zero-based sparse vector's non-zero indices.
- `sparseValue`
The `short[]` for the vector values of the non-zero indices.

Exceptions

`ArgumentOutOfRangeException` - `dimensions` index is less than zero; an `indices` element is less than zero or greater than or equal to `dimensions`; or a vector value is either too large or too small for a signed byte (`INT8`).

`ArgumentNullException` - The `indices` or `sparseValue` parameter is null.

`ArgumentException` - The `dimensions` are fewer than the length of `indices` or `sparseValue`, or the length of `indices` and `sparseValue` are not equal.

Remarks

The `ProviderType` is `OracleDbType.Vector_Int8`.

If the vector parameter has a length of zero, then the constructed instance will have null value and `IsNull` will be true.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSparseVector Class](#)
- [OracleSparseVector Members](#)

OracleSparseVector(dimensions, indices, sparseValue)

This constructor creates an instance of the `OracleSparseVector` class with a specified number of dimensions and non-zero vector values stored in a `float[]`.

Declaration

```
// C#  
public OracleSparseVector(long dimensions, long[] indices, float[] sparseValue);
```

Parameters

- **dimensions**
The number of sparse vector dimensions.
- **indices**
The `long[]` for the zero-based sparse vector's non-zero indices.
- **sparseValue**
The `float[]` for the vector values of the non-zero indices.

Exceptions

`ArgumentOutOfRangeException` – `dimensions` index is less than zero, or an element in `indices` is less than zero or greater than or equal to `dimensions`.

`ArgumentNullException` - The `indices` or `sparseValue` parameter is null.

`ArgumentException` - The `dimensions` are fewer than the length of `indices` or `sparseValue`, or the length of `indices` and `sparseValue` are not equal.

Remarks

The `ProviderType` is `OracleDbType.Vector_Float32`.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSparseVector Class](#)
- [OracleSparseVector Members](#)

OracleSparseVector(dimensions, indices, sparseValue)

This constructor creates an instance of the `OracleSparseVector` class with a specified number of dimensions and non-zero vector values stored in a `double[]`.

Declaration

```
// C#
public OracleSparseVector(long dimensions, long[] indices, double[] sparseValue);
```

Parameters

- **dimensions**
The number of sparse vector dimensions.
- **indices**
The long[] for the zero-based sparse vector's non-zero indices.
- **sparseValue**
The double[] for the vector values of the non-zero indices.

Exceptions

ArgumentOutOfRangeException – dimensions index is less than zero, or an element in indices is less than zero or greater than or equal to dimensions.

ArgumentNullException - The indices or sparseValue parameter is null.

ArgumentException - The dimensions are fewer than the length of indices or sparseValue, or the length of indices and sparseValue are not equal.

Remarks

The ProviderType is OracleDbType.Vector_Float64.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSparseVector Class](#)
- [OracleSparseVector Members](#)

OracleSparseVector Properties

OracleSparseVector properties are listed in [Table 6-206](#).

Table 6-206 OracleSparseVector Properties

Property	Description
Dimensions	Returns the number of dimensions of the OracleSparseVector
VectorType	Returns the vector type
ProviderType	Returns the OracleDbType value of the OracleSparseVector provider type
Value	Returns a .NET numeric array representing the sparse vector values at the non-zero indices

Table 6-206 (Cont.) OracleSparseVector Properties

Property	Description
Indices	Returns a <code>long[]</code> of the zero-based sparse vector's non-zero indices
Item	Obtains the element in an <code>OracleSparseVector</code> at the specified index
IsNull	Indicates whether the current instance has a null value

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSparseVector Class](#)
- [OracleSparseVector Members](#)

Dimensions

This property returns the number of dimensions of the `OracleSparseVector`.

Declaration

```
// C#  
public long Dimensions {get;}
```

Property Value

A long value

Exceptions

`OracleNullValueException` - The current instance has a null value.

`ObjectDisposedException` - The object is already disposed.

Remarks

Inherits `OracleVector` `Dimensions` property.

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSparseVector Class](#)
- [OracleSparseVector Members](#)

VectorType

This property returns the vector type.

Declaration

```
// C#  
public OracleVectorType VectorType {get;}
```

Property Value

OracleSparseVector always returns as type OracleVectorType.Sparse.

Exceptions

OracleNullValueException - The current instance has a null value.

ObjectDisposedException - The object is already disposed.

Remarks

Inherits OracleVector VectorType property.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSparseVector Class](#)
- [OracleSparseVector Members](#)

ProviderType

This property returns the OracleDbType value of the OracleSparseVector provider type.

Declaration

```
// C#  
public OracleDbType ProviderType {get;}
```

Property Value

The vector's provider type corresponds to its numeric format.

Possible values include:

- OracleDbType.Vector_Binary
- OracleDbType.Vector_Int8
- OracleDbType.Vector_Float32
- OracleDbType.Vector_Float64

Exceptions

OracleNullValueException - The current instance has a null value.

ObjectDisposedException - The object is already disposed.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSparseVector Class](#)
- [OracleSparseVector Members](#)

Value

This property returns a .NET numeric array representing the sparse vector values at the non-zero indices.

Declaration

```
// C#
public Array Value {get;}
```

Property Value

A .NET numeric array representing the sparse vector values at the non-zero indices. This array can be byte[], Int16[], float[], or double[] depending on the sparse vector's numeric format.

Exceptions

- OracleNullValueException - The current instance has a null value.
- ObjectDisposedException - The object is already disposed.

Remarks

This property's return type varies depending on the sparse vector's numeric format.

Numeric Format	Return Type
OracleDbType.Vector_Binary	byte[]
OracleDbType.Vector_Int8	short[]
OracleDbType.Vector_Float32	float[]
OracleDbType.Vector_Float64	double[]

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSparseVector Class](#)
- [OracleSparseVector Members](#)

Indices

This property returns a long[] of the zero-based sparse vector's non-zero indices.

Declaration

```
// C#  
public long[] Indices {get;}
```

Property Value

A long[] of the zero-based sparse vector's non-zero indices.

Exceptions

OracleNullValueException - The current instance has a null value.

ObjectDisposedException - The object is already disposed.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSparseVector Class](#)
- [OracleSparseVector Members](#)

Item

This property obtains the element in an OracleSparseVector at the specified index.

Declaration

```
// C#  
public object this [long index] {get;}
```

Parameters

index

The zero-based index of the element to get from the OracleSparseVector

Property Value

An object value of bool, short, float, or double at the specified index depending on the sparse vector's numeric format.

Exceptions

OracleNullValueException - The current instance has a null value.

ArgumentOutOfRangeException - Index is less than zero or index is equal to or greater than number of dimensions.

ObjectDisposedException - The object is already disposed.

Remarks

This property's return type varies depending on the sparse vector's numeric format.

Numeric Format	Return Type
OracleDbType.Vector_Binary	bool
OracleDbType.Vector_Int8	short
OracleDbType.Vector_Float32	float
OracleDbType.Vector_Float64	double

- ① **See Also**
- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
 - [OracleSparseVector Class](#)
 - [OracleSparseVector Members](#)

IsNull

This property indicates whether the current instance has a null value.

Declaration

```
// C#
public bool IsNull {get;}
```

Property Value

Returns true if the current instance has a null value; otherwise, returns false.

- ① **See Also**
- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
 - [OracleSparseVector Class](#)
 - [OracleSparseVector Members](#)

OracleSparseVector Instance Methods

OracleSparseVector instance methods are listed in [Table 6-207](#).

Table 6-207 OracleSparseVector Instance Methods

Instance Method	Description
ToInt16Array()	Returns a short[] value of the OracleSparseVector instance
ToFloatArray()	Returns a float[] value of the OracleSparseVector instance
ToDoubleArray()	Returns a double[] value of this OracleSparseVector instance
ToNonZeroInt16Array()	Returns a short[] of the sparse vector values at the non-zero indices

Table 6-207 (Cont.) OracleSparseVector Instance Methods

Instance Method	Description
ToNonZeroFloatArray()	Returns a <code>float[]</code> of the sparse vector values at the non-zero indices
ToNonZeroDoubleArray()	Returns a <code>double[]</code> of the sparse vector values at the non-zero indices
ToString()	Returns the <code>OracleSparseVector</code> string representation
Dispose()	Releases resources allocated by this object

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSparseVector Class](#)
- [OracleSparseVector Members](#)

ToInt16Array()

The method returns a `short[]` value of this `OracleSparseVector` instance.

Declaration

```
// C#  
public short[] ToInt16Array();
```

Return Value

A `short[]` representing the whole sparse vector in dense vector form.

Exceptions

`OracleNullValueException` - The current instance has a null value.

`ObjectDisposedException` - The object is already disposed.

Remarks

If the vector's numeric format is not `OracleDbType.Vector_Int8`, then this method implicitly converts the vector values to short types.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSparseVector Class](#)
- [OracleSparseVector Members](#)

ToFloatArray()

The method returns a `float[]` value of this `OracleSparseVector` instance.

Declaration

```
// C#  
public float[] ToFloatArray();
```

Return Value

A `float[]` representing the whole sparse vector in dense vector form.

Expections

`OracleNullValueException` - The current instance has a null value.

`ObjectDisposedException` - The object is already disposed.

Remarks

If the vector's numeric format is not `OracleDbType.Vector_Float32`, then this method implicitly converts the vector values to float types.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSparseVector Class](#)
- [OracleSparseVector Members](#)

ToDoubleArray()

The method returns a `double[]` value of this `OracleSparseVector` instance.

Declaration

```
// C#  
public double[] ToDoubleArray();
```

Return Value

A `double[]` representing the whole sparse vector in dense vector form.

Expections

`OracleNullValueException` - The current instance has a null value.

`ObjectDisposedException` - The object is already disposed.

Remarks

If the vector's numeric format is not `OracleDbType.Vector_Float64`, then this method implicitly converts the vector values to double types.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSparseVector Class](#)
- [OracleSparseVector Members](#)

ToNonZeroInt16Array()

The method returns a `short[]` of the sparse vector values at the non-zero indices.

Declaration

```
// C#  
public short[] ToNonZeroInt16Array();
```

Return Value

A `short[]` of the sparse vector values at the non-zero indices.

Expections

`OracleNullValueException` - The current instance has a null value.

`ObjectDisposedException` - The object is already disposed.

Remarks

If the vector's numeric format is not `OracleDbType.Vector_Int8`, then this method implicitly converts the vector values to short types.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSparseVector Class](#)
- [OracleSparseVector Members](#)

ToNonZeroFloatArray()

The method returns a `float[]` of the sparse vector values at the non-zero indices.

Declaration

```
// C#  
public float[] ToNonZeroFloatArray();
```

Return Value

A `float[]` of the sparse vector values at the non-zero indices.

Expections

`OracleNullValueException` - The current instance has a null value.

ObjectDisposedException - The object is already disposed.

Remarks

If the vector's numeric format is not `OracleDbType.Vector_Float32`, then this method implicitly converts the vector values to float types.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSparseVector Class](#)
- [OracleSparseVector Members](#)

ToNonZeroDoubleArray()

The method returns a `double[]` of the sparse vector values at the non-zero indices.

Declaration

```
// C#  
public double[] ToNonZeroDoubleArray();
```

Return Value

A `double[]` of the sparse vector values at the non-zero indices.

Exceptions

OracleNullValueException - The current instance has a null value.

ObjectDisposedException - The object is already disposed.

Remarks

If the vector's numeric format is not `OracleDbType.Vector_Float64`, then this method implicitly converts the vector values to double types.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSparseVector Class](#)
- [OracleSparseVector Members](#)

ToString()

This method returns the `OracleSparseVector` string representation

Declaration

```
// C#  
public string ToString();
```

Return Value

Returns the sparse vector representation as a JSON format string.

Expections

ObjectDisposedException - The object is already disposed.

Remarks

For BINARY sparse vectors, the JSON format string has 2 elements: dimensions and indices. For sparse vectors of all other numeric formats, the JSON format string has 3 elements: dimensions, indices, and sparse values.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSparseVector Class](#)
- [OracleSparseVector Members](#)

Dispose()

This instance method releases resources allocated by this object.

Declaration

```
// C#  
public void Dispose();
```

Implements

IDisposable

Remarks

The object cannot be reused after being disposed. Although some properties can still be accessed, their values cannot be accountable. Since resources are freed, method calls can lead to exceptions.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSparseVector Class](#)
- [OracleSparseVector Members](#)

OracleTransaction Class

An OracleTransaction object represents a local transaction.

Class Inheritance

System.Object
 System.MarshalByRefObject
 System.Data.Common.DbTransaction
 Oracle.DataAccess.Client.OracleTransaction

Declaration

```
// C#  
public sealed class OracleTransaction : DbTransaction  
  
// C#  
public sealed class OracleTransaction : MarshalByRefObject,  
    IDbTransaction, IDisposable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

The application calls `BeginTransaction` on the `OracleConnection` object to create an `OracleTransaction` object. The `OracleTransaction` object can be created in Read Committed mode only. Any other mode results in an exception.

The execution of a [DDL](#) statement in the context of a transaction is not recommended since it results in an implicit commit that is not reflected in the state of the `OracleTransaction` object.

All operations related to [savepoints](#) pertain to the current local transaction. Operations like commit and rollback performed on the transaction have no effect on data in any existing `DataSet`.

Example

```
// Database Setup, if you have not done so yet.  
/*  
connect scott/tiger@oracle  
DROP TABLE MyTable;  
CREATE TABLE MyTable (MyColumn NUMBER);  
--CREATE TABLE MyTable (MyColumn NUMBER PRIMARY KEY);
```

```
*/

// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;

class OracleTransactionSample
{
    static void Main()
    {
        // Drop & Create MyTable as indicated Database Setup, at beginning

        // This sample starts a transaction and inserts two records with the same
        // value for MyColumn into MyTable.
        // If MyColumn is not a primary key, the transaction will commit.
        // If MyColumn is a primary key, the second insert will violate the
        // unique constraint and the transaction will rollback.

        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleCommand cmd = con.CreateCommand();

        // Check the number of rows in MyTable before transaction
        cmd.CommandText = "SELECT COUNT(*) FROM MyTable";
        int myTableCount = int.Parse(cmd.ExecuteScalar().ToString());

        // Print the number of rows in MyTable
        Console.WriteLine("myTableCount = " + myTableCount);

        // Start a transaction
        OracleTransaction txn = con.BeginTransaction(
            IsolationLevel.ReadCommitted);

        try
        {
            // Insert the same row twice into MyTable
            cmd.CommandText = "INSERT INTO MyTable VALUES (1)";
            cmd.ExecuteNonQuery();
            cmd.ExecuteNonQuery(); // This may throw an exception
            txn.Commit();
        }
        catch (Exception e)
        {
            // Print the exception message
            Console.WriteLine("e.Message = " + e.Message);

            // Rollback the transaction
            txn.Rollback();
        }

        // Check the number of rows in MyTable after transaction
        cmd.CommandText = "SELECT COUNT(*) FROM MyTable";
        myTableCount = int.Parse(cmd.ExecuteScalar().ToString());

        // Prints the number of rows
        // If MyColumn is not a PRIMARY KEY, the value should increase by two.
    }
}
```

```
// If MyColumn is a PRIMARY KEY, the value should remain same.
Console.WriteLine("myTableCount = " + myTableCount);

txn.Dispose();
cmd.Dispose();

con.Close();
con.Dispose();
}
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleTransaction Members](#)
- [OracleTransaction Static Methods](#)
- [OracleTransaction Properties](#)

OracleTransaction Members

OracleTransaction members are listed in the following tables.

OracleTransaction Static Methods

The OracleTransaction static method is listed in [Table 6-208](#).

Table 6-208 OracleTransaction Static Method

Method	Description
Equals	Inherited from System.Object (Overloaded)

OracleTransaction Properties

OracleTransaction properties are listed in [Table 6-209](#).

Table 6-209 OracleTransaction Properties

Property	Description
IsolationLevel	Specifies the isolation level for the transaction
Connection	Specifies the connection that is associated with the transaction

OracleTransaction Public Methods

OracleTransaction public methods are listed in [Table 6-210](#).

Table 6-210 OracleTransaction Public Methods

Public Method	Description
Commit	Commits the database transaction

Table 6-210 (Cont.) OracleTransaction Public Methods

Public Method	Description
CreateObjRef	Inherited from System.MarshalByRefObject
Dispose	Frees the resources used by the OracleTransaction object
Equals	Inherited from System.Object (Overloaded)
GetHashCode	Inherited from System.Object
GetLifetimeService	Inherited from System.MarshalByRefObject
GetType	Inherited from System.Object
InitializeLifetimeService	Inherited from System.MarshalByRefObject
Rollback	Rolls back a database transaction (Overloaded)
Save	Creates a savepoint within the current transaction
ToString	Inherited from System.Object

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleTransaction Class](#)

OracleTransaction Static Methods

The OracleTransaction static method is listed in [Table 6-211](#).

Table 6-211 OracleTransaction Static Method

Method	Description
Equals	Inherited from System.Object (Overloaded)

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleTransaction Class](#)
- [OracleTransaction Members](#)

OracleTransaction Properties

OracleTransaction properties are listed in [Table 6-212](#).

Table 6-212 OracleTransaction Properties

Property	Description
IsolationLevel	Specifies the isolation level for the transaction
Connection	Specifies the connection that is associated with the transaction

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleTransaction Class](#)
- [OracleTransaction Members](#)

IsolationLevel

This property specifies the isolation level for the transaction.

Declaration

```
// C#  
public override IsolationLevel IsolationLevel {get;}
```

Property Value

IsolationLevel

Implements

IDbTransaction

Exceptions

InvalidOperationException - The transaction has already completed.

Remarks

Default = IsolationLevel.ReadCommitted

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleTransaction Class](#)
- [OracleTransaction Members](#)

Connection

This property specifies the connection that is associated with the transaction.

Declaration

```
// C#
public OracleConnection Connection {get;}
```

Property Value

Connection

Implements

IDbTransaction

Remarks

This property indicates the OracleConnection object that is associated with the transaction.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleTransaction Class](#)
- [OracleTransaction Members](#)

OracleTransaction Public Methods

OracleTransaction public methods are listed in [Table 6-213](#).

Table 6-213 OracleTransaction Public Methods

Public Method	Description
Commit	Commits the database transaction
CreateObjRef	Inherited from System.MarshalByRefObject
Dispose	Frees the resources used by the OracleTransaction object
Equals	Inherited from System.Object (Overloaded)
GetHashCode	Inherited from System.Object
GetLifetimeService	Inherited from System.MarshalByRefObject
GetType	Inherited from System.Object
InitializeLifetimeService	Inherited from System.MarshalByRefObject
Rollback	Rolls back a database transaction (Overloaded)
Save	Creates a savepoint within the current transaction
ToString	Inherited from System.Object

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleTransaction Class](#)
- [OracleTransaction Members](#)

Commit

This method commits the database transaction.

Declaration

```
// C#  
public override void Commit();
```

Implements

IDbTransaction

Exceptions

InvalidOperationException - The transaction has already been completed successfully, has been rolled back, or the associated connection is closed.

Remarks

Upon a successful commit, the transaction enters a completed state.

Upon `Commit()`, `AutoCommit` and `IsolationLevel` return to the respective values specified on the `OracleConnection` object.

Example

```
// Database Setup, if you have not done so yet  
/*  
connect scott/tiger@oracle  
DROP TABLE MyTable;  
CREATE TABLE MyTable (MyColumn NUMBER);  
--CREATE TABLE MyTable (MyColumn NUMBER PRIMARY KEY);  
  
*/  
  
// C#  
  
using System;  
using System.Data;  
using Oracle.DataAccess.Client;  
  
class CommitSample  
{  
    static void Main()  
    {  
        // Drop & Create MyTable as indicated in Database Setup, at beginning  
  
        // This sample starts a transaction and inserts two records with the same  
        // value for MyColumn into MyTable.  
        // If MyColumn is not a primary key, the transaction will commit.
```

```
// If MyColumn is a primary key, the second insert will violate the
// unique constraint and the transaction will rollback.

string constr = "User Id=scott;Password=tiger;Data Source=oracle";
OracleConnection con = new OracleConnection(constr);
con.Open();

OracleCommand cmd = con.CreateCommand();

// Check the number of rows in MyTable before transaction
cmd.CommandText = "SELECT COUNT(*) FROM MyTable";
int myTableCount = int.Parse(cmd.ExecuteScalar().ToString());

// Print the number of rows in MyTable
Console.WriteLine("myTableCount = " + myTableCount);

// Start a transaction
OracleTransaction txn = con.BeginTransaction(
    IsolationLevel.ReadCommitted);

try
{
    // Insert the same row twice into MyTable
    cmd.CommandText = "INSERT INTO MyTable VALUES (1)";
    cmd.ExecuteNonQuery();
    cmd.ExecuteNonQuery(); // This may throw an exception
    txn.Commit();
}
catch (Exception e)
{
    // Print the exception message
    Console.WriteLine("e.Message = " + e.Message);

    // Rollback the transaction
    txn.Rollback();
}

// Check the number of rows in MyTable after transaction
cmd.CommandText = "SELECT COUNT(*) FROM MyTable";
myTableCount = int.Parse(cmd.ExecuteScalar().ToString());

// Prints the number of rows
// If MyColumn is not a PRIMARY KEY, the value should increase by two.
// If MyColumn is a PRIMARY KEY, the value should remain same.
Console.WriteLine("myTableCount = " + myTableCount);

txn.Dispose();
cmd.Dispose();

con.Close();
con.Dispose();
}
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleTransaction Class](#)
- [OracleTransaction Members](#)

Dispose

This method frees the resources used by the OracleTransaction object.

Declaration

```
// C#  
public void Dispose();
```

Implements

IDisposable

Remarks

This method releases both the managed and unmanaged resources held by the OracleTransaction object. If the transaction is not in a completed state, an attempt to rollback the transaction is made.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleTransaction Class](#)
- [OracleTransaction Members](#)

Rollback

Rollback rolls back a database transaction.

Overload List:

- [Rollback\(\)](#)
This method rolls back a database transaction.
- [Rollback\(string\)](#)
This method rolls back a database transaction to a [savepoint](#) within the current transaction.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleTransaction Class](#)
- [OracleTransaction Members](#)

Rollback()

This method rolls back a database transaction.

Declaration

```
// C#  
public override void Rollback();
```

Implements

IDbTransaction

Exceptions

InvalidOperationException - The transaction has already been completed successfully, has been rolled back, or the associated connection is closed.

Remarks

After a Rollback(), the OracleTransaction object can no longer be used because the Rollback ends the transaction.

Upon Rollback(), AutoCommit and IsolationLevel return to the respective values specified on the OracleConnection object.

Example

```
// Database Setup, if you have not done so yet.  
/*  
connect scott/tiger@oracle  
DROP TABLE MyTable;  
CREATE TABLE MyTable (MyColumn NUMBER);  
  
*/  
  
// C#  
  
using System;  
using System.Data;  
using Oracle.DataAccess.Client;  
  
class RollbackSample  
{  
    static void Main()  
    {  
        // Drop & Create MyTable as indicated previously in Database Setup  
  
        // This sample starts a transaction and inserts one record into MyTable.  
        // It then rollsback the transaction, the number of rows remains the same
```

```
string constr = "User Id=scott;Password=tiger;Data Source=oracle";
OracleConnection con = new OracleConnection(constr);
con.Open();

OracleCommand cmd = con.CreateCommand();

// Check the number of rows in MyTable before transaction
cmd.CommandText = "SELECT COUNT(*) FROM MyTable";
int myTableCount = int.Parse(cmd.ExecuteScalar().ToString());

// Print the number of rows in MyTable
Console.WriteLine("myTableCount = " + myTableCount);

// Start a transaction
OracleTransaction txn = con.BeginTransaction(
    IsolationLevel.ReadCommitted);

// Insert a row into MyTable
cmd.CommandText = "INSERT INTO MyTable VALUES (1)";
cmd.ExecuteNonQuery();

// Rollback the transaction
txn.Rollback();

// Check the number of rows in MyTable after transaction
cmd.CommandText = "SELECT COUNT(*) FROM MyTable";
myTableCount = int.Parse(cmd.ExecuteScalar().ToString());

// Prints the number of rows, should remain the same
Console.WriteLine("myTableCount = " + myTableCount);

txn.Dispose();
cmd.Dispose();

con.Close();
con.Dispose();
}
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleTransaction Class](#)
- [OracleTransaction Members](#)

Rollback(string)

This method rolls back a database transaction to a [savepoint](#) within the current transaction.

Declaration

```
// C#
public override void Rollback(string savepointName);
```

Parameters

- *savepointName*

The name of the savepoint to rollback to, in the current transaction.

Exceptions

`InvalidOperationException` - The transaction has already been completed successfully, has been rolled back, or the associated connection is closed.

Remarks

After a rollback to a savepoint, the current transaction remains active and can be used for further operations.

The *savepointName* specified does not have to match the case of the *savepointName* created using the `Save` method, since savepoints are created in the database in a case-insensitive manner.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleTransaction Class](#)
- [OracleTransaction Members](#)

Save

This method creates a [savepoint](#) within the current transaction.

Declaration

```
// C#  
public void Save(string savepointName);
```

Parameters

- *savepointName*
The name of the savepoint being created in the current transaction.

Exceptions

`InvalidOperationException` - The transaction has already been completed.

Remarks

After creating a savepoint, the transaction does not enter a completed state and can be used for further operations.

The *savepointName* specified is created in the database in a case-insensitive manner. Calling the `Rollback` method rolls back to *savepointName*. This allows portions of a transaction to be rolled back, instead of the entire transaction.

Example

```
// Database Setup, if you have not done so yet.  
/*  
connect scott/tiger@oracle  
DROP TABLE MyTable;
```



```
CREATE TABLE MyTable (MyColumn NUMBER);

*/

// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;

class SaveSample
{
    static void Main()
    {
        // Drop & Create MyTable as indicated in Database Setup, at beginning

        // This sample starts a transaction and creates a savepoint after
        // inserting one record into MyTable.
        // After inserting the second record it rollsback to the savepoint
        // and commits the transaction. Only the first record will be inserted

        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleCommand cmd = con.CreateCommand();

        // Check the number of rows in MyTable before transaction
        cmd.CommandText = "SELECT COUNT(*) FROM MyTable";
        int myTableCount = int.Parse(cmd.ExecuteScalar().ToString());

        // Print the number of rows in MyTable
        Console.WriteLine("myTableCount = " + myTableCount);

        // Start a transaction
        OracleTransaction txn = con.BeginTransaction(
            IsolationLevel.ReadCommitted);

        // Insert a row into MyTable
        cmd.CommandText = "INSERT INTO MyTable VALUES (1)";
        cmd.ExecuteNonQuery();

        // Create a savepoint
        txn.Save("MySavePoint");

        // Insert another row into MyTable
        cmd.CommandText = "insert into mytable values (2)";
        cmd.ExecuteNonQuery();

        // Rollback to the savepoint
        txn.Rollback("MySavePoint");

        // Commit the transaction
        txn.Commit();

        // Check the number of rows in MyTable after transaction
        cmd.CommandText = "SELECT COUNT(*) FROM MyTable";
        myTableCount = int.Parse(cmd.ExecuteScalar().ToString());

        // Prints the number of rows, should have increased by 1
        Console.WriteLine("myTableCount = " + myTableCount);
    }
}
```

```
        txn.Dispose();  
        cmd.Dispose();  
  
        con.Close();  
        con.Dispose();  
    }  
}
```

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleTransaction Class](#)
- [OracleTransaction Members](#)

OracleVector Class

An OracleVector object represents a dense Oracle VECTOR. It is the base class for all Oracle VECTOR types.

Class Inheritance

System.Object

Oracle.DataAccess.Types.OracleVector

Declaration

```
// C#  
public class OracleVector : INullable, IXmlSerializable, IDisposable
```

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [Artificial Intelligence Vectors and Semantic Search](#)
- [OracleVector Members](#)
- [OracleVector Constructors](#)
- [OracleVector Properties](#)
- [IOracleVector Instance Methods](#)

OracleVector Members

OracleVector members are listed in the following tables.

OracleVector Constructors

The OracleVector constructors is listed in [Table 6-214](#).

Table 6-214 OracleVector Constructors

Constructor	Description
OracleVector Constructors	OracleVector constructors instantiate new instances of OracleVector class. (Overloaded)

OracleVector Properties

OracleVector properties are listed in [Table 6-215](#).

Table 6-215 OracleVector Properties

Property	Description
Dimensions	Returns the number of dimensions of the OracleVector
VectorType	Returns the vector's storage format
ProviderType	Returns the OracleDbType value of the OracleVector provider type
Value	Returns a .NET array representing the vector
Item[long]	Obtains the element in an OracleVector at the specified index
IsNull	bool IsNull returns true if the current instance has a null value; otherwise, returns false.

OracleVector Static Field

OracleVector static fields are listed in [Table 6-216](#).

Table 6-216 OracleVector Static Field

Static Field	Description
Null	static readonly OracleVector Null represents a null value that can be assigned to an OracleVector instance

OracleVector Instance Methods

OracleVector Instance methods are listed in [Table 6-217](#).

Table 6-217 OracleVector Instance Methods

Instance Method	Description
ToInt16Array()	Returns a short[] value of the OracleVector instance
ToFloatArray()	Returns a float[] value of the OracleVector instance
ToDoubleArray()	Returns a double[] value of this OracleVector instance
ToByteArray()	Returns a byte[] value of this OracleVector instance
ToString()	Returns the OracleVector string representation
Dispose()	Releases resources allocated by this object

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleVector Class](#)

OracleVector Constructors

OracleVector constructors instantiate new instances of OracleVector class.

Overload List:

- [OracleVector\(vectorString\)](#)
This constructor creates an instance of the OracleVector class with the string representation of a vector.
- [OracleVector\(vectorString, providerType\)](#)
This constructor creates an instance of the OracleVector class with the string representation of a vector.
- [OracleVector\(vectorValues\)](#)
This constructor creates an instance of the OracleVector class with a vector in the short[].
- [OracleVector\(vectorValues\)](#)
This constructor creates an instance of the OracleVector class with a vector in the float[].

- [OracleVector\(vectorValues\)](#)

This constructor creates an instance of the `OracleVector` class with a vector in the `double[]`.

- [OracleVector\(vectorValues\)](#)

This constructor creates an instance of the `OracleVector` class with a vector in the `byte[]`.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleVector Class](#)
- [OracleVector Members](#)

OracleVector(vectorString)

This constructor creates an instance of the `OracleVector` class with the string representation of a vector.

Declaration

```
// C#  
public OracleVector(string vectorString);
```

Parameters

vectorString

The string representation of a vector

Exceptions

`ArgumentException` - The string parameter is an invalid string representation of an `OracleVector`.

`ArgumentNullException` - The string parameter is null.

Remarks

The `ProviderType` is `OracleDbType.Vector_Float32`.

If the parameter value is an empty string, then the constructed instance will have null value and `IsNull` will be true.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleVector Class](#)
- [OracleVector Members](#)

OracleVector(vectorString, providerType)

This constructor creates an instance of the `OracleVector` class with the string representation of a vector.

Declaration

```
// C#  
public OracleVector(string vectorString, OracleDbType providerType);
```

Parameters

- *vectorString*
The string representation of a vector
- *providerType*
The `OracleDbType` of the resulting `OracleVector`.

Exceptions

`ArgumentException` - The vector string parameter is an invalid string representation of an `OracleVector` or the `providerType` is not a `OracleDbType.Vector*`

`ArgumentNullException` - The string parameter is null.

Remarks

The allowed `providerType` values are:

- `OracleDbType.Vector`
- `OracleDbType.Vector_Int8`
- `OracleDbType.Vector_Float32`
- `OracleDbType.Vector_Float64`
- `OracleDbType.Vector_Binary`

If `providerType` is `OracleDbType.Vector`, then the `OracleSparseVector` `ProviderType` will be `OracleDbType.Vector_Float32`.

If the parameter value is an empty string, then the constructed instance will have null value and `IsNull` will be true.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleVector Class](#)
- [OracleVector Members](#)

OracleVector(vectorValues)

This constructor creates an instance of the `OracleVector` class with a vector in the `short[]`.

Declaration

```
// C#  
public OracleVector(short[] vectorValues);
```

Parameters

vectorValues

A `short[]` that represents a vector.

Exceptions

`ArgumentNullException` - The parameter is null.

`ArgumentOutOfRangeException` - A value in the parameter vector is either too large or too small for a signed byte (`INT8`).

Remarks

The `ProviderType` is `OracleDbType.Vector_Int8`.

If the vector parameter has a length of zero, then the constructed instance will have null value and `IsNull` will be true.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleVector Class](#)
- [OracleVector Members](#)

OracleVector(vectorValues)

This constructor creates an instance of the `OracleVector` class with a vector in the `float[]`.

Declaration

```
// C#  
public OracleVector(float[] vectorValues);
```

Parameters

vectorValues

A `float[]` that represents a vector.

Exceptions

`ArgumentNullException` - The parameter is null.

Remarks

The `ProviderType` is `OracleDbType.Vector_Float32`.

If the vector parameter has a length of zero, then the constructed instance will have null value and `IsNull` will be true.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleVector Class](#)
- [OracleVector Members](#)

OracleVector(vectorValues)

This constructor creates an instance of the `OracleVector` class with a vector in the `double[]`.

Declaration

```
// C#  
public OracleVector(double[] vectorValues);
```

Parameters

vectorValues

A `double[]` that represents a vector.

Exceptions

`ArgumentNullException` - The parameter is null.

Remarks

The `ProviderType` is `OracleDbType.Vector_Float64`.

If the vector parameter has a length of zero, then the constructed instance will have null value and `IsNull` will be true.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleVector Class](#)
- [OracleVector Members](#)

OracleVector(vectorValues)

This constructor creates an instance of the `OracleVector` class with a vector in the `byte[]`.

Declaration

```
// C#  
public OracleVector(byte[] vectorValues);
```

Parameters

vectorValues

A `byte[]` that represents a vector with BINARY numeric format.

Exceptions

ArgumentNullException - The parameter is null.

Remarks

The ProviderType is OracleDbType.Vector_Binary.

If the vector parameter has a length of zero, then the constructed instance will have null value and IsNull will be true.

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleVector Class](#)
- [OracleVector Members](#)

OracleVector Properties

OracleVector properties are listed in [Table 6-218](#).

Table 6-218 OracleVector Properties

Property	Description
Dimensions	Returns the number of dimensions of the OracleVector
VectorType	Returns the vector's storage format
ProviderType	Returns the OracleDbType value of the OracleVector provider type
Value	Returns a .NET array representing the vector
Item[long]	Obtains the element in an OracleVector at the specified index
IsNull	bool IsNull returns true if the current instance has a null value; otherwise, returns false.

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleVector Class](#)
- [OracleVector Members](#)

Dimensions

This property returns the number of dimensions of the OracleVector.

Declaration

```
// C#  
public long Dimensions {get;}
```

Property Value

A long value

Exceptions

OracleNullValueException - The current instance has a null value.

ObjectDisposedException - The object is already disposed.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleVector Class](#)
- [OracleVector Members](#)

VectorType

This property returns the vector's storage format.

Declaration

```
// C#  
public OracleVectorType VectorType {get;}
```

Property Value

The OracleVectorType representing the storage format of the OracleVector.

Exceptions

OracleNullValueException - The current instance has a null value.

ObjectDisposedException - The object is already disposed.

Remarks

OracleVector will always return the vector value in dense format regardless of whether the vector is dense or sparse. To retrieve sparse values, cast the OracleVector to OracleSparseVector and use the OracleSparseVector APIs.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleVector Class](#)
- [OracleVector Members](#)

ProviderType

This property returns the `OracleDbType` value of the `OracleVector` provider type.

Declaration

```
// C#  
public OracleDbType ProviderType {get;}
```

Property Value

The vector's provider type corresponds to its numeric format.

Possible values:

- `OracleDbType.Vector_Int8`
- `OracleDbType.Vector_Float32`
- `OracleDbType.Vector_Float64`
- `OracleDbType.Vector_Binary`

Exceptions

`OracleNullValueException` - The current instance has a null value.

`ObjectDisposedException` - The object is already disposed.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleVector Class](#)
- [OracleVector Members](#)

Value

This property returns a .NET array representing the vector.

Declaration

```
// C#  
public Array Value {get;}
```

Property Value

A .NET array representing the vector. This array can be `Int16[]`, `float[]`, `double[]`, or `byte[]` depending on the vector numeric format.

Exceptions

`OracleNullValueException` - The current instance has a null value.

`ObjectDisposedException` - The object is already disposed.

Remarks

The property return type varies based on the vector numeric format.

Numeric Format	Return Type
OracleDbType.Vector_Int8	short[]
OracleDbType.Vector_Float32	float[]
OracleDbType.Vector_Float64	double[]
OracleDbType.Vector_Binary	byte[]

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleVector Class](#)
- [OracleVector Members](#)

Item[long]

This property obtains the element in an OracleVector at the specified index.

Declaration

```
// C#
public object this [long index] {get;}
```

Parameters

index

The zero-based index of the element to retrieve from the OracleVector

Property Value

A short, float, double, or bool object at the specified index depending on the vector numeric format.

Exceptions

OracleNullValueException - The current instance has a null value.

ArgumentOutOfRangeException - Index is less than zero or index is equal to or greater than vector dimensions.

ObjectDisposedException - The object is already disposed.

Remarks

The property return type varies based on the vector numeric format.

Numeric Format	Return Type
OracleDbType.Vector_Int8	short

Numeric Format	Return Type
OracleDbType.Vector_Float32	float
OracleDbType.Vector_Float64	double
OracleDbType.Vector_Binary	bool

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleVector Class](#)
- [OracleVector Members](#)

OracleVector Instance Methods

OracleVector instance methods are listed in [Table 6-219](#).

Table 6-219 OracleVector Instance Methods

Instance Method	Description
ToInt16Array()	Returns a short[] value of the OracleVector instance
ToFloatArray()	Returns a float[] value of the OracleVector instance
ToDoubleArray()	Returns a double[] value of this OracleVector instance
ToByteArray()	Returns a byte[] value of this OracleVector instance
ToString()	Returns the OracleVector string representation
Dispose()	Releases resources allocated by this object

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleVector Class](#)
- [OracleVector Members](#)

ToInt16Array()

The method returns a short[] value of the OracleVector instance.

Declaration

```
// C#
public short[] ToInt16Array();
```

Return Value

A short[] representing the vector

Exceptions

InvalidCastException - The vector numeric format is binary.

OracleNullValueException - The current instance has a null value.

ObjectDisposedException - The object is already disposed.

Remarks

If the vector's numeric format is not `OracleDbType.Vector_Int8`, then this method implicitly converts the vector values to short types.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleVector Class](#)
- [OracleVector Members](#)

ToFloatArray()

The method returns a `float[]` value of the `OracleVector` instance.

Declaration

```
// C#  
public float[] ToFloatArray();
```

Return Value

A `float[]` representing the vector

Exceptions

InvalidCastException - The vector numeric format is binary.

OracleNullValueException - The current instance has a null value.

ObjectDisposedException - The object is already disposed.

Remarks

If the vector's numeric format is not `OracleDbType.Vector_Float32`, then this method implicitly converts the vector values to float types.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleVector Class](#)
- [OracleVector Members](#)

ToDoubleArray()

The method returns a `double[]` value of this `OracleVector` instance.

Declaration

```
// C#  
public double[] ToDoubleArray();
```

Return Value

A `double[]`

Exceptions

`InvalidCastException` - The vector numeric format is binary.

`OracleNullValueException` - The current instance has a null value.

`ObjectDisposedException` - The object is already disposed.

Remarks

If the vector's numeric format is not `OracleDbType.Vector_Float64`, then this method implicitly converts the vector values to double types.

❶ See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleVector Class](#)
- [OracleVector Members](#)

ToByteArray()

The method returns a `byte[]` value of this `OracleVector` instance.

Declaration

```
// C#  
public byte[] ToByteArray();
```

Return Value

A `byte[]`

Exceptions

`OracleNullValueException` - The current instance has a null value.

`ObjectDisposedException` - The object is already disposed.

Remarks

If the vector's numeric format is not `OracleDbType.Vector_Binary`, then this method implicitly converts the vector values to byte types.

This is the only valid `To*Array()` method for Binary format vectors. Binary sparse vectors are returned as dense vector byte arrays.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleVector Class](#)
- [OracleVector Members](#)

ToString()

Returns the `OracleVector` string representation.

Declaration

```
// C#  
public string ToString();
```

Return Value

Returns the vector representation as a JSON format string.

Remarks

If the vector is sparse, then `OracleVector.ToString()` will return a JSON string in dense format.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleVector Class](#)
- [OracleVector Members](#)

Dispose()

This instance method releases resources allocated by this object.

Declaration

```
// C#  
public void Dispose();
```

Implements

`IDisposable`

Remarks

The object cannot be reused after being disposed. Although some properties can still be accessed, their values cannot be accountable. Since resources are freed, further method calls can lead to exceptions.

❶ See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleVector Class](#)
- [OracleVector Members](#)

OracleAllowedLogonVersionClient Enumeration

OracleAllowedLogonVersionClient enumerated values define minimum authentication protocol that is to be used for a given OracleConnection object or for a given application (depending on where it's set).

[OracleAllowedLogonVersionClient Enumeration Values](#) lists all the OracleAllowedLogonVersionClient enumeration values with a description of each enumerated value.

Table 6-220 OracleAllowedLogonVersionClient Enumeration Values

Member Name	Description
Version12a	equivalent to 12a setting for SQLNET.ALLOWED_LOGON_VERSION_CLIENT
Version12	equivalent to 12 setting for SQLNET.ALLOWED_LOGON_VERSION_CLIENT
Version11	equivalent to 11 setting for SQLNET.ALLOWED_LOGON_VERSION_CLIENT
Version10	equivalent semantically to Version11
Version9	equivalent semantically to Version11
Version8	equivalent semantically to Version11

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

❶ See Also

- *Oracle Database Net Services Reference* for more information
- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)

OracleCollectionType Enumeration

OracleCollectionType enumerated values specify whether or not the OracleParameter object represents a collection, and if so, specifies the collection type.

[Table 6-221](#) lists all the OracleCollectionType enumeration values with a description of each enumerated value.

Table 6-221 OracleCollectionType Enumeration Values

Member Name	Description
None	Is not a collection type
PLSQLAssociativeArray	Indicates that the collection type is a PL/SQL Associative Array (or PL/SQL Index-By Table)

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- "[OracleParameter Class](#)"
- "[CollectionType](#)"

OracleConnectionOpenReason Enumeration

This enumeration's values specify the reason for triggering a ConnectionOpen event. Applications can retrieve the reason through the OracleConnectionOpenEventArgs Reason property

[Table 6-222](#) lists all the OracleConnectionOpenReason enumeration values with a description of each enumerated value.

Table 6-222 OracleConnectionOpenReason Enumeration Values

Member Name	Description
HardConnect	Indicates a hard connect to the database. A new connection has been created.
SoftConnect	Indicates a soft connect to the database. A connection from the pool has been dispensed.

A connection created for Application Continuity replay has an enumeration value of HardConnect.

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- "[OracleParameter Class](#)"
- "[CollectionType](#)"

OracleConnectionType Enumeration

OracleConnectionType enumerated values specify whether a particular connection object is associated with an Oracle database connection, a TimesTen database connection, or no physical connection at all.

[Table 6-223](#) lists all the OracleConnectionType enumeration values with a description of each enumerated value.

Table 6-223 OracleConnectionType Enumeration Values

Member Name	Description
Undefined	No connection is associated with the OracleConnection object.
Oracle	The OracleConnection object is associated with an Oracle database.
TimesTen	The OracleConnection object is associated with a TimesTen database.

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	Oracle.DataAccess.dll
Namespace	Oracle.DataAccess.Client
.NET Framework	See System Requirements

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- "[OracleConnection Class](#)"
- "[ConnectionType](#)"

OracleDBAPrivilege Enumeration

OracleDBAPrivilege enumerated values are used to explicitly specify the DBA Privilege required while creating the OracleCredential object.

[Table 6-223](#) lists all the OracleDBAPrivilege enumeration values with a description of each enumerated value.

Table 6-224 OracleDBAPrivilege Enumeration Values

Member Name	Description
None	Default value, no DBA privilege set.
SYSASM	SYSASM connection
SYSBACKUP	SYSBACKUP connection
SYSDBA	SYSDBA connection
SYSDG	SYSDG connection
SYSKM	SYSKM connection
SYSOPER	SYSOPER connection
SYSRAC	SYSRAC connection

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
.NET (Core)	-	-	See System Requirements

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleCredential Class](#)
- [DBAPrivilege](#)

OracleDBShutdownMode Enumeration

OracleDBShutdownMode enumerated values specify the database shutdown options.

[Table 6-226](#) lists all the OracleDBShutdownMode enumeration values with a description of each enumerated value.

Table 6-225 OracleDBShutdownMode Enumeration Values

Member Name	Description
Default	Refuses new connections and waits for existing connections to end.
Transactional	Refuses new connections and does not allow any new transactions. Waits for active transactions to commit.
TransactionalLocal	Refuses new connections and does not allow any new transactions. Waits for only local transactions to commit.
Immediate	Does not wait for current calls to complete or users to disconnect from the database. All uncommitted transactions are terminated and rolled back.
Final	Shuts down the database. Used in the second call for shutdown after the database has been closed and dismounted.
Abort	Does not wait for current calls to complete or users to disconnect from the database. All uncommitted transactions are terminated and are not rolled back.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
.NET (Core)	-	-	See System Requirements

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- "[OracleConnection Class](#)"
- "[Shutdown](#)"

OracleDBStartupMode Enumeration

OracleDBStartupMode enumerated values specify the database startup options.

[Table 6-226](#) lists all the OracleDBStartupMode enumeration values with a description of each enumerated value.

Table 6-226 OracleDBStartupMode Enumeration Values

Member Name	Description
NoRestriction	Starts the database and allows access to all users.
Restrict	Starts the database and allows database access only to users having the CREATE SESSION and RESTRICTED SESSION privileges. These privileges are normally assigned to database administrators.
Force	Shuts down a running instance in abort mode and starts a new instance.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- ["OracleConnection Class"](#)
- ["Startup"](#)

OracleDbType Enumeration

OracleDbType enumerated values are used to explicitly specify the OracleDbType of an OracleParameter.

[Table 6-227](#) lists all the OracleDbType enumeration values with a description of each enumerated value.

Table 6-227 OracleDbType Enumeration Values

Member Name	Description
Array	Oracle Collection (VArray or Nested Table)
BFile	Oracle BFILE type
BinaryFloat	Oracle BINARY_FLOAT type
BinaryDouble	Oracle BINARY_DOUBLE type
Blob	Oracle BLOB type
Boolean	Oracle BOOLEAN type
Byte	byte type
Char	Oracle CHAR type
Clob	Oracle CLOB type
Date	Oracle DATE type
Decimal	Oracle NUMBER type
Double	8-byte FLOAT type
Int16	2-byte INTEGER type
Int32	4-byte INTEGER type
Int64	8-byte INTEGER type
IntervalDS	Oracle INTERVAL DAY TO SECOND type
IntervalYM	Oracle INTERVAL YEAR TO MONTH type
Long	Oracle LONG type
Json	Oracle JSON type
LongRaw	Oracle LONG RAW type
NChar	Oracle NCHAR type
NClob	Oracle NCLOB type
NVarChar2	Oracle NVARCHAR2 type
Object	Oracle Object
Raw	Oracle RAW type

Table 6-227 (Cont.) OracleDbType Enumeration Values

Member Name	Description
Ref	Oracle REF
RefCursor	Oracle REF CURSOR type
Single	4-byte FLOAT type, supports 6 precisions
TimeStamp	Oracle TIMESTAMP type
TimeStampLTZ	Oracle TIMESTAMP WITH LOCAL TIME ZONE type
TimeStampTZ	Oracle TIMESTAMP WITH TIME ZONE type
Varchar2	Oracle VARCHAR2 type
Vector	Numeric format type vector. Either INT8, FLOAT32, or FLOAT64
Vector_Int8	INT8 numeric format type vector
Vector_Float32	FLOAT32 numeric format type vector
Vector_Float64	FLOAT64 numeric format type vector
XmlType	Oracle XMLType type

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- "[OracleParameter Class](#)"
- "[OracleParameterCollection Class](#)"
- OracleParameter "[OracleDbType](#)"

OracleDRCPurity Enumeration

OracleDRCPurity enumerated values specify the session purity.

[Table 6-228](#) lists all the OracleDRCPurity enumeration values with a description of each enumerated value.

Table 6-228 OracleDRCPPurity Enumeration Values

Member Name	Description
New	The application requires a session without any previously set session state.
Pooled	An application can reuse a pooled session (i.e. the session can have been used before)

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- "[OracleConnection Class](#)"
- [DRCPPurity](#)

OracleIdentityType Enumeration

The `OracleIdentityType` enumeration specifies how Oracle identity column values are generated.

[Table 6-229](#) lists all the `OracleIdentityType` enumeration values with a description of each enumerated value.

Table 6-229 OracleIdentityType Members

Member Name	Description
GeneratedAlways	Indicates that unique values are generated for every insertion. No updates or inserts are allowed for this identity column.
GeneratedByDefault	Indicates that the values are generated only if no explicit value is provided for the identity column. Null values are not allowed for this identity column.
GeneratedByDefaultOnNull	Indicates that the values are generated only if no explicit value is provided or a NULL is inserted for the identity column.

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	Oracle.DataAccess.dll
Namespace	Oracle.DataAccess.Client
.NET Framework	See System Requirements

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- "[OracleDataAdapter Class](#) "
- OracleDataAdapter "[IdentityInsert](#)"
- OracleDataAdapter "[IdentityUpdate](#)"

OracleParameterStatus Enumeration

The OracleParameterStatus enumeration type indicates whether a NULL value is fetched from a column, or truncation has occurred during the fetch, or a NULL value is to be inserted into a database column.

[Table 6-230](#) lists all the OracleParameterStatus enumeration values with a description of each enumerated value.

Table 6-230 OracleParameterStatus Members

Member Name	Description
Success	Indicates that (for input parameters) the input value has been assigned to the column. For output parameter, it indicates that the provider assigned an intact value to the parameter.
NullFetched	Indicates that a NULL value has been fetched from a column or an OUT parameter.
NullInsert	Indicates that a NULL value is to be inserted into a column.
Truncation	Indicates that truncation has occurred when fetching the data from the column.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
.NET (Core)	-	-	See System Requirements

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- "[OracleParameter Class](#)"
- OracleParameter "[ArrayBindStatus](#) "
- OracleParameter "[Value](#) "

OraclePasswordAuth Enumeration

OraclePasswordAuth enumerated values specify which PASSWORD_AUTH mode must be enabled for database token authentication. There is no impact on external authentication. Possible values are:

[Table 6-231](#) lists all the OracleTokenAuth enumeration values with a description of each enumerated value.

Table 6-231 OraclePasswordAuth Members

Member Name	Description
PasswordVerifier	Default value. Indicates database authentication is to be used.
OciToken	ODP.NET makes REST calls to IAM endpoint to retrieve the database token for authentication.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAccessToken Class](#)
- [OracleRefreshAccessTokenEventArgs Class](#)

OracleSessionlessTransactionStartBehavior Enumeration

OracleSessionlessTransactionStartBehavior enumerated values specify when to start a sessionless transaction.

[Table 6-232](#) lists all the OracleSessionlessTransactionStartBehavior enumeration values with a description of each enumerated value.

Table 6-232 OracleSessionlessTransactionStartBehavior Members

Member Name	Description
StartBeforeExecution	When OracleCommand invokes the command, the sessionless transaction is started or resumed first and the SQL is then executed in one server round-trip.
StartImmediate	Start or resume the sessionless transaction immediately.

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleSessionlessTransactionOptions Class](#)

OracleTokenAuth Enumeration

OracleTokenAuth enumerated values are used to explicitly specify which TOKEN_AUTH mode needs to be enabled for token authentication. Possible values for this enumeration are:

[Table 6-233](#) lists all the OracleTokenAuth enumeration values with a description of each enumerated value.

Table 6-233 OracleTokenAuth Members

Member Name	Description
Disabled	Default value. Token authentication is DISABLED.
OciApiKey	Token authentication using OCI_API_KEY authentication flow. <i>Not available in ODP.NET, Unmanaged Driver</i>
OciInstancePrincipal	Token authentication using OCI_INSTANCE_PRINCIPAL authentication flow. <i>Not available in ODP.NET, Unmanaged Driver</i>
OciDelegationToken	Token authentication using OCI_DELEGATION_TOKEN authentication flow. <i>Not available in ODP.NET, Unmanaged Driver</i>
OciInteractive	Token authentication using OCI_INTERACTIVE authentication flow. <i>Not available in ODP.NET, Unmanaged Driver</i>
OciDefault	Token authentication using the most appropriate method depending on the application environment. ODP.NET first attempts to retrieve the token using the OCI API key. If that fails, then ODP.NET checks whether it is in a Cloud Shell environment to attempt to get the token using delegation token authentication flow. Lastly, it attempts to use the instance principal authentication flow. <i>Not available in ODP.NET, Unmanaged Driver</i>
OciToken	Token authentication is enabled for Oracle Identity and Access Management.
OAuth	Token authentication enabled for Microsoft Entra ID.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAccessToken Class](#)
- [OracleRefreshAccessTokenEventArgs Class](#)

OracleVectorType Enumeration

OracleVectorType enumerated values specify the vector storage format. Possible values are: [Table 6-234](#) lists all the OracleVectorType enumeration values with a description of each enumerated value.

Table 6-234 OracleVectorType Members

Member Name	Description
Dense	The vector is stored in the dense format.
Sparse	The vector is stored in the dense format.

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- OracleVector Class
- OracleSparseVector Class

Oracle Data Provider for .NET XML-Related Classes

This chapter describes ODP.NET XML-related classes and enumerations.

This chapter contains these topics:

- [OracleXmlCommandType Enumeration](#)
- [OracleXmlQueryProperties Class](#)
- [OracleXmlSaveProperties Class](#)
- [OracleXmlStream Class](#)
- [OracleXmlType Class](#)

All offsets are 0-based for OracleXmlStream object parameters.

OracleXmlCommandType Enumeration

The OracleXmlCommandType enumeration specifies the values that are allowed for the XmlCommandType property of the OracleCommand class. It is used to specify the type of XML operation.

[Table 7-1](#) lists all the OracleXmlCommandType enumeration values with a description of each enumerated value.

Table 7-1 OracleXmlCommandType Members

Member Name	Description
None	No XML operation is desired
Query	The command text is a SQL query and the result of the query is an XML document. The SQL query needs to be a select statement
Insert	The command text is an XML document containing rows to insert.
Update	The command text is an XML document containing rows to update.
Delete	The command text is an XML document containing rows to delete.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
.NET (Core)	-	-	See System Requirements

 **See Also**
"[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"

OracleXmlQueryProperties Class

An OracleXmlQueryProperties object represents the XML properties used by the OracleCommand class when the XmlCommandType property is Query.

Class Inheritance

System.Object
 System.OracleXmlQueryProperties

Declaration

public sealed class OracleXmlQueryProperties : ICloneable

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

OracleXmlQueryProperties can be accessed, and modified using the XmlQueryProperties property of the OracleCommand class. Each OracleCommand object has its own instance of the OracleXmlQueryProperties class in the XmlQueryProperties property.

Use the default constructor to get a new instance of the OracleXmlQueryProperties. Use the OracleXmlQueryProperties.Clone() method to get a copy of an OracleXmlQueryProperties instance.

Example

This example retrieves relational data as XML.

```
// C#

using System;
using System.IO;
using System.Data;
using System.Xml;
using System.Text;
using Oracle.DataAccess.Client;

class OracleXmlQueryPropertiesSample
{
    static void Main()
    {
        int rows = 0;
        StreamReader sr = null;

        // Define the XSL document for doing the transform.
        string xslstr = "<?xml version='1.0'?>\n" +
            "<xsl:stylesheet version='1.0'\n" +
            "  xmlns:xsl='http://www.w3.org/1999/XSL/Transform'\n" +
            "  <xsl:output encoding='utf-8'/>\n" +
            "  <xsl:template match='/'>\n" +
            "    <EMPLOYEES>\n" +
            "      <xsl:apply-templates select='ROWSET'/>\n" +
            "    </EMPLOYEES>\n" +
            "  </xsl:template>\n" +
            "  <xsl:template match='ROWSET'\n" +
            "    <xsl:apply-templates select='ROW'/>\n" +
            "  </xsl:template>\n" +
            "  <xsl:template match='ROW'\n" +
            "    <EMPLOYEE>\n" +
            "      <EMPLOYEE_ID>\n" +
            "        <xsl:apply-templates select='EMPNO'/>\n" +
            "      </EMPLOYEE_ID>\n" +
            "      <EMPLOYEE_NAME>\n" +
            "        <xsl:apply-templates select='ENAME'/>\n" +
            "      </EMPLOYEE_NAME>\n" +
            "      <HIRE_DATE>\n" +
            "        <xsl:apply-templates select='HIREDATE'/>\n" +
            "      </HIRE_DATE>\n" +
            "      <JOB_TITLE>\n" +
            "        <xsl:apply-templates select='JOB'/>\n" +
            "      </JOB_TITLE>\n" +
            "    </EMPLOYEE>\n" +
            "  </xsl:template>\n" +
            "</xsl:stylesheet>\n";

        // Create the connection.
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        // Set the date, and timestamp formats for Oracle 9i Release 2, or later.
        // This is just needed for queries.
        if (!con.ServerVersion.StartsWith("9.0") &&
            !con.ServerVersion.StartsWith("8.1"))
        {
            OracleGlobalization sessionParams = con.GetSessionInfo();
```

```
        sessionParams.DateFormat = "YYYY-MM-DD\"T\"HH24:MI:SS";
        sessionParams.TimestampFormat = "YYYY-MM-DD\"T\"HH24:MI:SS.FF3";
        sessionParams.TimestampTZFormat = "YYYY-MM-DD\"T\"HH24:MI:SS.FF3";
        con.SetSessionInfo(sessionParams);
    }

    // Create the command.
    OracleCommand cmd = new OracleCommand("", con);

    // Set the XML command type to query.
    cmd.XmlCommandType = OracleXmlCommandType.Query;

    // Set the SQL query.
    cmd.CommandText = "select * from emp e where e.empno = :empno";

    // Set command properties that affect XML query behaviour.
    cmd.BindByName = true;

    // Bind values to the parameters in the SQL query.
    Int32 empNum = 7369;
    cmd.Parameters.Add("empno", OracleDbType.Int32, empNum,
        ParameterDirection.Input);

    // Set the XML query properties.
    cmd.XmlQueryProperties.MaxRows = 1;
    cmd.XmlQueryProperties.RootTag = "ROWSET";
    cmd.XmlQueryProperties.RowTag = "ROW";
    cmd.XmlQueryProperties.Xslt = xslstr;

    // Test query execution without returning a result.
    Console.WriteLine("SQL query: select * from emp e where e.empno = 7369");
    Console.WriteLine("Maximum rows: all rows (-1)");
    Console.WriteLine("Return Value from OracleCommand.ExecuteNonQuery():");
    rows = cmd.ExecuteNonQuery();
    Console.WriteLine(rows);
    Console.WriteLine("\n");

    // Get the XML document as an XmlReader.
    Console.WriteLine("SQL query: select * from emp e where e.empno = 7369");
    Console.WriteLine("Maximum rows: all rows (-1)");
    Console.WriteLine("XML Document from OracleCommand.ExecuteXmlReader():");

    XmlReader xmlReader = cmd.ExecuteXmlReader();
    XmlDocument xmlDocument = new XmlDocument();
    xmlDocument.PreserveWhitespace = true;
    xmlDocument.Load(xmlReader);
    Console.WriteLine(xmlDocument.OuterXml);
    Console.WriteLine("\n");

    // Change the SQL query, and set the maximum number of rows to 2.
    cmd.CommandText = "select * from emp e";
    cmd.Parameters.Clear();
    cmd.XmlQueryProperties.MaxRows = 2;

    // Get the XML document as a Stream.
    Console.WriteLine("SQL query: select * from emp e");
    Console.WriteLine("Maximum rows: 2");
    Console.WriteLine("XML Document from OracleCommand.ExecuteStream():");
    Stream stream = cmd.ExecuteStream();
    sr = new StreamReader(stream, Encoding.Unicode);
    Console.WriteLine(sr.ReadToEnd());
    Console.WriteLine("\n");
```

```
// Get all the rows.
cmd.XmlQueryProperties.MaxRows = -1;

// Append the XML document to an existing Stream.
Console.WriteLine("SQL query: select * from emp e");
Console.WriteLine("Maximum rows: all rows (-1)");
Console.WriteLine("XML Document from OracleCommand.ExecuteToStream():");
MemoryStream mstream = new MemoryStream(32);
cmd.ExecuteToStream(mstream);
mstream.Seek(0, SeekOrigin.Begin);
sr = new StreamReader(mstream, Encoding.Unicode);
Console.WriteLine(sr.ReadToEnd());
Console.WriteLine("\n");

// Clean up.
cmd.Dispose();
con.Close();
con.Dispose();
}
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleXmlQueryProperties Members](#)
- [OracleXmlQueryProperties Constructor](#)
- [OracleXmlQueryProperties Properties](#)
- [OracleXmlQueryProperties Public Methods](#)

OracleXmlQueryProperties Members

OracleXmlQueryProperties members are listed in the following tables.

OracleXmlQueryProperties Constructors

The OracleXmlQueryProperties constructors are listed in [Table 7-2](#).

Table 7-2 OracleXmlQueryProperties Constructors

Constructor	Description
OracleXmlQueryProperties Constructor	Instantiates a new instance of the OracleXmlQueryProperties class

OracleXmlQueryProperties Properties

The OracleXmlQueryProperties properties are listed in [Table 7-3](#).

Table 7-3 OracleXmlQueryProperties Properties

Name	Description
MaxRows	Specifies the maximum number of rows from the result set of the query that can be represented in the result XML document
RootTag	Specifies the root element of the result XML document
RowTag	Specifies the value of the XML element which identifies a row of data from the result set in an XML document
Xslt	Specifies the XSL document used for XML transformation using XSLT
XsltParams	Specifies parameters for the XSL document

OracleXmlQueryProperties Public Methods

The OracleXmlQueryProperties public methods are listed in [Table 7-4](#).

Table 7-4 OracleXmlQueryProperties Public Methods

Name	Description
Clone	Creates a copy of an OracleXmlQueryProperties object

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleXmlQueryProperties Class](#)

OracleXmlQueryProperties Constructor

The OracleXmlQueryProperties constructor instantiates a new instance of the OracleXmlQueryProperties class.

Declaration

```
// C#  
public OracleXmlQueryProperties();
```

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleXmlQueryProperties Class](#)
- [OracleXmlQueryProperties Members](#)

OracleXmlQueryProperties Properties

The OracleXmlQueryProperties properties are listed in [Table 7-5](#).

Table 7-5 OracleXmlQueryProperties Properties

Name	Description
MaxRows	Specifies the maximum number of rows from the result set of the query that can be represented in the result XML document
RootTag	Specifies the root element of the result XML document
RowTag	Specifies the value of the XML element which identifies a row of data from the result set in an XML document
Xslt	Specifies the XSL document used for XML transformation using XSLT
XsltParams	Specifies parameters for the XSL document

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleXmlQueryProperties Class](#)
- [OracleXmlQueryProperties Members](#)

MaxRows

This property specifies the maximum number of rows from the result set of the query that can be represented in the result XML document.

Declaration

```
// C#  
public int MaxRows {get; set;}
```

Property Value

The maximum number of rows.

Exceptions

ArgumentException - The new value for MaxRows is not valid.

Remarks

Default value is -1.

Possible values are:

- -1 (all rows).
- A number greater than or equal to 0.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleXmlQueryProperties Class](#)
- [OracleXmlQueryProperties Members](#)

RootTag

This property specifies the root element of the result XML document.

Declaration

```
// C#  
public string RootTag {get; set;}
```

Property Value

The root element of the result XML document.

Remarks

The default root tag is ROWSET.

To indicate that no root tag is to be used in the result XML document, set this property to null or "" or String.Empty.

If both RootTag and RowTag are set to null, an XML document is returned only if the result set returns one row and one column.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleXmlQueryProperties Class](#)
- [OracleXmlQueryProperties Members](#)

RowTag

This property specifies the value of the XML element which identifies a row of data from the result set in an XML document.

Declaration

```
// C#  
public string RowTag {get; set;}
```

Property Value

The value of the XML element.

Remarks

The default is ROW.

To indicate that no row tag is be used in the result XML document, set this property to null or "" or String.Empty.

If both RootTag and RowTag are set to null, an XML document is returned only if the result set returns one row and one column.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleXmlQueryProperties Class](#)
- [OracleXmlQueryProperties Members](#)

Xslt

This property specifies the XSL document used for XML transformation using XSLT.

Declaration

```
// C#  
public string Xslt {get; set;}
```

Property Value

The XSL document used for XML transformation.

Remarks

Default value is null.

The XSL document is used for XML transformation of the XML document generated from the result set of the query.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleXmlQueryProperties Class](#)
- [OracleXmlQueryProperties Members](#)

XsltParams

This property specifies parameters for the XSL document.

Declaration

```
// C#  
public string XsltParams {get; set;}
```

Property Value

The parameters for the XSL document.

Remarks

Default value is null.

The parameters are specified as a string of "name=value" pairs of the form "param1=value1; param2=value2; ..." delimited by semicolons.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleXmlQueryProperties Class](#)
- [OracleXmlQueryProperties Members](#)

OracleXmlQueryProperties Public Methods

The OracleXmlQueryProperties public methods are listed in [Table 7-6](#).

Table 7-6 OracleXmlQueryProperties Public Methods

Name	Description
Clone	Creates a copy of an OracleXmlQueryProperties object

Clone

This method creates a copy of an OracleXmlQueryProperties object.

Declaration

```
// C#
public object Clone();
```

Return Value

An OracleXmlQueryProperties object

Implements

ICloneable

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleXmlQueryProperties Class](#)
- [OracleXmlQueryProperties Members](#)

OracleXmlSaveProperties Class

An OracleXmlSaveProperties object represents the XML properties used by the OracleCommand class when the XmlCommandType property is Insert, Update, or Delete.

Class Inheritance

System.Object

System.OracleXmlSaveProperties

Declaration

```
public sealed class OracleXmlSaveProperties : ICloneable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

OracleXmlSaveProperties can be accessed and modified using the XmlSaveProperties property of the OracleCommand class. Each OracleCommand object has its own instance of the OracleXmlSaveProperties class in the XmlSaveProperties property.

Use the default constructor to get a new instance of OracleXmlSaveProperties. Use the OracleXmlSaveProperties.Clone() method to get a copy of an OracleXmlSaveProperties instance.

Example

This sample demonstrates how to do inserts, updates, and deletes to a relational table or view using an XML document.

```
// C#
/* -- If HR account is being locked, you need to log on as a DBA
   -- to unlock the account first. Unlock a locked users account:

   ALTER USER hr ACCOUNT UNLOCK;
*/

using System;
```

```

using Oracle.DataAccess.Client;

class OracleXmlSavePropertiesSample
{
    static void Main()
    {
        string[] KeyColumnsList = null;
        string[] UpdateColumnsList = null;
        int rows = 0;

        // Create the connection.
        string constr = "User Id=hr;Password=hr;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        // Create the command.
        OracleCommand cmd = new OracleCommand("", con);

        // Set the XML command type to insert.
        cmd.XmlCommandType = OracleXmlCommandType.Insert;

        // Set the XML document.
        cmd.CommandText = "<?xml version='1.0'?'>\n" +
            "<ROWSET>\n" +
            "  <MYROW num = \"1\">\n" +
            "    <EMPLOYEE_ID>1234</EMPLOYEE_ID>\n" +
            "    <LAST_NAME>Smith</LAST_NAME>\n" +
            "    <EMAIL>Smith@Oracle.com</EMAIL>\n" +
            "    <HIRE_DATE>1982-01-23T00:00:00.000</HIRE_DATE>\n" +
            "    <JOB_ID>IT_PROG</JOB_ID>\n" +
            "  </MYROW>\n" +
            "  <MYROW num = \"2\">\n" +
            "    <EMPLOYEE_ID>1235</EMPLOYEE_ID>\n" +
            "    <LAST_NAME>Barney</LAST_NAME>\n" +
            "    <EMAIL>Barney@Oracle.com</EMAIL>\n" +
            "    <HIRE_DATE>1982-01-23T00:00:00.000</HIRE_DATE>\n" +
            "    <JOB_ID>IT_PROG</JOB_ID>\n" +
            "  </MYROW>\n" +
            "</ROWSET>\n";

        // Set the XML save properties.
        KeyColumnsList = new string[1];
        KeyColumnsList[0] = "EMPLOYEE_ID";
        UpdateColumnsList = new string[5];
        UpdateColumnsList[0] = "EMPLOYEE_ID";
        UpdateColumnsList[1] = "LAST_NAME";
        UpdateColumnsList[2] = "EMAIL";
        UpdateColumnsList[3] = "HIRE_DATE";
        UpdateColumnsList[4] = "JOB_ID";
        cmd.XmlSaveProperties.KeyColumnsList = KeyColumnsList;
        cmd.XmlSaveProperties.RowTag = "MYROW";
        cmd.XmlSaveProperties.Table = "employees";
        cmd.XmlSaveProperties.UpdateColumnsList = UpdateColumnsList;
        cmd.XmlSaveProperties.Xslt = null;
        cmd.XmlSaveProperties.XsltParams = null;

        // Do the inserts.
        rows = cmd.ExecuteNonQuery();
        Console.WriteLine("rows: " + rows);

        // Set the XML command type to update.
        cmd.XmlCommandType = OracleXmlCommandType.Update;
    }
}

```

```

// Set the XML document.
cmd.CommandText = "<?xml version=\"1.0\"?>\n" +
    "<ROWSET>\n" +
    "  <MYROW num = \"1\">\n" +
    "    <EMPLOYEE_ID>1234</EMPLOYEE_ID>\n" +
    "    <LAST_NAME>Adams</LAST_NAME>\n" +
    "  </MYROW>\n" +
    "</ROWSET>\n";

// Set the XML save properties.
KeyColumnsList = new string[1];
KeyColumnsList[0] = "EMPLOYEE_ID";
UpdateColumnsList = new string[1];
UpdateColumnsList[0] = "LAST_NAME";
cmd.XmlSaveProperties.KeyColumnsList = KeyColumnsList;
cmd.XmlSaveProperties.UpdateColumnsList = UpdateColumnsList;
rows = cmd.ExecuteNonQuery();
Console.WriteLine("rows: " + rows);

// Set the XML command type to delete.
cmd.XmlCommandType = OracleXmlCommandType.Delete;

// Set the XML document.
cmd.CommandText = "<?xml version=\"1.0\"?>\n" +
    "<ROWSET>\n" +
    "  <MYROW num = \"1\">\n" +
    "    <EMPLOYEE_ID>1234</EMPLOYEE_ID>\n" +
    "  </MYROW>\n" +
    "  <MYROW num = \"2\">\n" +
    "    <EMPLOYEE_ID>1235</EMPLOYEE_ID>\n" +
    "  </MYROW>\n" +
    "</ROWSET>\n";

// Set the XML save properties.
KeyColumnsList = new string[1];
KeyColumnsList[0] = "EMPLOYEE_ID";
cmd.XmlSaveProperties.KeyColumnsList = KeyColumnsList;
cmd.XmlSaveProperties.UpdateColumnsList = null;

// Do the deletes.
rows = cmd.ExecuteNonQuery();
Console.WriteLine("rows: " + rows);

// Clean up.
cmd.Dispose();
con.Close();
con.Dispose();
}
}

```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleXmlSaveProperties Members](#)
- [OracleXmlSaveProperties Constructor](#)
- [OracleXmlSaveProperties Properties](#)
- [OracleXmlSaveProperties Public Methods](#)

OracleXmlSaveProperties Members

OracleXmlSaveProperties members are listed in the following tables.

OracleXmlSaveProperties Constructor

OracleXmlSaveProperties constructors are listed in [Table 7-7](#)

Table 7-7 OracleXmlSaveProperties Constructor

Constructor	Description
OracleXmlSaveProperties Constructor	Instantiates a new instance of the OracleXmlSaveProperties class

OracleXmlSaveProperties Properties

The OracleXmlSaveProperties properties are listed in [Table 7-8](#).

Table 7-8 OracleXmlSaveProperties Properties

Name	Description
KeyColumnsList	Specifies the list of columns used as a key to locate existing rows for update or delete using an XML document
RowTag	Specifies the value for the XML element that identifies a row of data in an XML document
Table	Specifies the name of the table or view to which changes are saved
UpdateColumnsList	Specifies the list of columns to update or insert
Xslt	Specifies the XSL document used for XML transformation using XSLT
XsltParams	Specifies the parameters for the XSLT document specified in the Xslt property

OracleXmlSaveProperties Public Methods

The OracleXmlSaveProperties public methods are listed in [Table 7-9](#).

Table 7-9 OracleXmlSaveProperties Public Methods

Name	Description
Clone	Creates a copy of an OracleXmlSaveProperties object

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleXmlSaveProperties Class](#)
- [OracleXmlSaveProperties Members](#)

OracleXmlSaveProperties Constructor

The OracleXmlSaveProperties constructor instantiates a new instance of OracleXmlSaveProperties class.

Declaration

```
// C#  
public OracleXmlSaveProperties;
```

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleXmlSaveProperties Class](#)
- [OracleXmlSaveProperties Members](#)

OracleXmlSaveProperties Properties

The OracleXmlSaveProperties properties are listed in [Table 7-10](#).

Table 7-10 OracleXmlSaveProperties Properties

Name	Description
KeyColumnsList	Specifies the list of columns used as a key to locate existing rows for update or delete using an XML document
RowTag	Specifies the value for the XML element that identifies a row of data in an XML document
Table	Specifies the name of the table or view to which changes are saved
UpdateColumnsList	Specifies the list of columns to update or insert
Xslt	Specifies the XSL document used for XML transformation using XSLT
XsltParams	Specifies the parameters for the XSLT document specified in the Xslt property

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleXmlSaveProperties Class](#)
- [OracleXmlSaveProperties Members](#)

KeyColumnsList

This property specifies the list of columns used as a key to locate existing rows for update or delete using an XML document.

Declaration

```
// C#  
public string[] KeyColumnsList {get; set;}
```

Property Value

The list of columns.

Remarks

Default value is null.

The first null value (if any) terminates the list.

KeyColumnsList usage with XMLCommandType property values:

- Insert - KeyColumnsList is ignored and can be null.
- Update - KeyColumnsList must be specified; it identifies the columns to use to find the rows to be updated.
- Delete - If KeyColumnsList is null, all the column values in each row element in the XML document are used to locate the rows to delete. Otherwise, KeyColumnsList specifies the columns used to identify the rows to delete.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleXmlSaveProperties Class](#)
- [OracleXmlSaveProperties Members](#)

RowTag

This property specifies the value for the XML element that identifies a row of data in an XML document.

Declaration

```
// C#  
public string RowTag {get; set;}
```

Property Value

An XML element name.

Remarks

The default value is ROW.

Each element in the XML document identifies one row in a table or view.

If RowTag is set to "" or null, no row tag is used in the XML document. In this case, the XML document is assumed to contain only one row.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleXmlSaveProperties Class](#)
- [OracleXmlSaveProperties Members](#)

Table

This property specifies the name of the table or view to which changes are saved.

Declaration

```
// C#  
public string Table {get; set;}
```

Property Value

A table name.

Remarks

Default value is null.

The property must be set to a valid table or view name.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleXmlSaveProperties Class](#)
- [OracleXmlSaveProperties Members](#)

UpdateColumnsList

This property specifies the list of columns to update or insert.

Declaration

```
// C#  
public string[] UpdateColumnsList {get; set;}
```

Property Value

A list of columns.

Remarks

Default value is null.

The first null value (if any) terminates the list.

UpdateColumnList usage with XMLCommandType property values:

- Insert - UpdateColumnList indicates which columns are assigned values when a new row is created. If UpdateColumnList is null, then all columns are assigned values. If a column is on the UpdateColumnList, but no value is specified for the row in the XML file, then NULL is used. If a column is not on the UpdateColumnList, then the default value for that column is used.
- Update - UpdateColumnList specifies columns to modify for each row of data in the XML document. If UpdateColumnList is null, all the values in each XML element in the XML document are used to modify the columns.
- Delete - UpdateColumnsList is ignored and can be null.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleXmlSaveProperties Class](#)
- [OracleXmlSaveProperties Members](#)

Xslt

This property specifies the XSL document used for XML transformation using XSLT.

Declaration

```
// C#  
public string Xslt {get; set;}
```

Property Value

The XSL document used for XML transformation.

Remarks

Default = null.

The XSL document is used for XSLT transformation of a given XML document. The transformed XML document is used to save changes to the table or view.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleXmlSaveProperties Class](#)
- [OracleXmlSaveProperties Members](#)

XsltParams

This property specifies the parameters for the XSLT document specified in the Xslt property.

Declaration

```
// C#
public string XsltParams {get; set;}
```

Property Value

The parameters for the XSLT document.

Remarks

Default is null.
This property is a string delimited by semicolons in "name=value" pairs of the form "param1=value1; param2=value2; ...".

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleXmlSaveProperties Class](#)
- [OracleXmlSaveProperties Members](#)

OracleXmlSaveProperties Public Methods

The OracleXmlSaveProperties public methods are listed in [Table 7-11](#).

Table 7-11 OracleXmlSaveProperties Public Methods

Name	Description
Clone	Creates a copy of an OracleXmlSaveProperties object

Clone

This method creates a copy of an OracleXmlSaveProperties object.

Declaration

```
// C#
public object Clone();
```

Return Value

An OracleXmlSaveProperties object

Implements

ICloneable

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleXmlSaveProperties Class](#)
- [OracleXmlSaveProperties Members](#)

OracleXmlStream Class

An OracleXmlStream object represents a read-only stream of XML data stored in an OracleXmlType object.

Class Inheritance

System.Object
 System.MarshalByRefObject
 System.Stream
 System.OracleXmlStream

Declaration

```
// C#
public sealed class OracleXmlStream : IDisposable, ICloneable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleXmlStream Members](#)
- [OracleXmlStream Constructor](#)
- [OracleXmlStream Static Methods](#)
- [OracleXmlStream Instance Properties](#)
- [OracleXmlStream Instance Methods](#)

OracleXmlStream Members

OracleXmlStream members are listed in the following tables.

OracleXmlStream Constructors

The OracleXmlStream constructors are listed in [Table 7-12](#).

Table 7-12 OracleXmlStream Constructors

Constructor	Description
OracleXmlStream Constructor	Creates an instance of an OracleXmlStream object which provides a Stream representation of the XML data stored in an OracleXmlType

OracleXmlStream Static Methods

The OracleXmlStream static methods are listed in [Table 7-13](#).

Table 7-13 OracleXmlStream Static Methods

Methods	Description
Equals	Inherited from System.Object (Overloaded)

OracleXmlStream Instance Properties

The OracleXmlStream instance properties are listed in [Table 7-14](#).

Table 7-14 OracleXmlStream Instance Properties

Properties	Description
CanRead	Indicates whether or not the XML stream can be read

Table 7-14 (Cont.) OracleXmlStream Instance Properties

Properties	Description
CanSeek	Indicates whether or not forward and backward seek operation can be performed
CanWrite	OracleXmlStream is a sequential read only, forward only stream.
Connection	Indicates the OracleConnection that is used to retrieve the XML data
Length	Indicates the number of bytes in the XML stream
Position	Gets or sets the byte position within the stream
Value	Returns the XML data, starting from the first character in the stream as a string

OracleXmlStream Instance Methods

The OracleXmlStream instance methods are listed in [Table 7-15](#).

Table 7-15 OracleXmlStream Instance Methods

Methods	Description
BeginRead	Inherited from System.IO.Stream
BeginWrite	Inherited from System.IO.Stream
Clone	Creates a copy of an OracleXmlStream object
Close	Closes the current stream and releases any resources associated with it
Dispose	Releases resources allocated by this object
EndRead	Inherited from System.IO.Stream
EndWrite	Inherited from System.IO.Stream
Equals	Inherited from System.Object
Flush	<i>Not Supported</i>
GetHashCode	Inherited from System.Object
GetLifetimeService	Inherited from System.MarshalByRefObject
GetType	Inherited from System.Object
InitializeLifetimeService	Inherited from System.MarshalByRefObject
Read	Reads a specified amount from the current stream instance and populates the array buffer (Overloaded)
ReadAsync	Returns a Task-based asynchronous version of OracleXmlStream.Read(), which reads a specified amount of data and populates the array buffer (Overloaded)
ReadByte	Inherited from System.IO.Stream
Seek	Sets the position within the current stream and returns the new position within the current stream
SetLength	<i>Not Supported</i>
ToString	Inherited from System.Object
Write	<i>Not Supported</i>

Table 7-15 (Cont.) OracleXmlStream Instance Methods

Methods	Description
WriteByte	<i>Not Supported</i>
WriteLine	<i>Not Supported</i>

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleXmlStream Class](#)

OracleXmlStream Constructor

This constructor creates an instance of an `OracleXmlStream` object which provides a `Stream` representation of the XML data stored in an `OracleXmlType` object.

Declaration

```
// C#  
public OracleXmlStream(OracleXmlType xmlType);
```

Parameters

- *xmlType*
The `OracleXmlType` object.

Remarks

The `OracleXmlStream` implicitly uses the `OracleConnection` object from the `OracleXmlType` object from which it was constructed.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleXmlStream Class](#)
- [OracleXmlStream Members](#)

OracleXmlStream Static Methods

The `OracleXmlStream` static methods are listed in [Table 7-16](#).

Table 7-16 OracleXmlStream Static Methods

Methods	Description
Equals	Inherited from <code>System.Object</code> (Overloaded)

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleXmlStream Class](#)
- [OracleXmlStream Members](#)

OracleXmlStream Instance Properties

The `OracleXmlStream` instance properties are listed in [Table 7-17](#).

Table 7-17 OracleXmlStream Instance Properties

Properties	Description
CanRead	Indicates whether or not the XML stream can be read
CanSeek	Indicates whether or not forward and backward seek operation can be performed
CanWrite	<code>OracleXmlStream</code> is a sequential read only, forward only stream.
Connection	Indicates the <code>OracleConnection</code> that is used to retrieve the XML data
Length	Indicates the number of bytes in the XML stream
Position	Gets or sets the byte position within the stream
Value	Returns the XML data, starting from the first character in the stream as a string

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleXmlStream Class](#)
- [OracleXmlStream Members](#)

CanRead

Overrides `Stream`

This property indicates whether or not the XML stream can be read.

Declaration

```
// C#  
public override bool CanRead{get;}
```

Property Value

If the XML stream is can be read, returns true; otherwise, returns false.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlStream Class](#)
- [OracleXmlStream Members](#)

CanSeek

Overrides Stream

This property indicates whether or not forward and backward seek operation can be performed.

Declaration

```
// C#  
public override bool CanSeek{get;}
```

Property Value

If forward and backward seek operations can be performed, this property returns true. Otherwise, returns false.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlStream Class](#)
- [OracleXmlStream Members](#)

CanWrite

Overrides Stream

OracleXmlStream is a sequential read only, forward only stream.

Declaration

```
// C#  
public override bool CanWrite{get;}
```

Property Value

Always returns false.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlStream Class](#)
- [OracleXmlStream Members](#)

Connection

This instance property indicates the `OracleConnection` that is used to retrieve the XML data.

Declaration

```
// C#  
public OracleConnection Connection {get;}
```

Property Value

An `OracleConnection`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlStream Class](#)
- [OracleXmlStream Members](#)

Length

Overrides `Stream`

This property indicates the number of bytes in the XML stream.

Declaration

```
// C#  
public override Int64 Length{get;}
```

Property Value

An `Int64` value representing the number of bytes in the XML stream. An empty stream has a length of 0 bytes.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleXmlStream Class](#)
- [OracleXmlStream Members](#)

Position

Overrides Stream

This property gets or sets the byte position within the stream.

Declaration

```
// C#  
public override Int64 Position{get; set;}
```

Property Value

An Int64 that indicates the current position in the stream.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

ArgumentOutOfRangeException - The Position is less than 0.

Remarks

The beginning of the stream is represented by position 0. Seeking to any location beyond the length of the stream is supported.

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleXmlStream Class](#)
- [OracleXmlStream Members](#)

Value

This property returns the XML data, starting from the first character of the stream as a string.

Declaration

```
// C#
public string Value{get; set;}
```

Property Value

A string.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The value of `Position` is neither used nor changed by using this property.

The maximum length of the string that can be returned by this property is 2 GB.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlStream Class](#)
- [OracleXmlStream Members](#)

OracleXmlStream Instance Methods

The `OracleXmlStream` instance methods are listed in [Table 7-18](#).

Table 7-18 OracleXmlStream Instance Methods

Methods	Description
<code>BeginRead</code>	Inherited from <code>System.IO.Stream</code>
<code>BeginWrite</code>	Inherited from <code>System.IO.Stream</code>
Clone	Creates a copy of an <code>OracleXmlStream</code> object
Close	Closes the current stream and releases any resources associated with it
Dispose	Releases resources allocated by this object
<code>EndRead</code>	Inherited from <code>System.IO.Stream</code>
<code>EndWrite</code>	Inherited from <code>System.IO.Stream</code>
<code>Equals</code>	Inherited from <code>System.Object</code>

Table 7-18 (Cont.) OracleXmlStream Instance Methods

Methods	Description
Flush	<i>Not Supported</i>
GetHashCode	Inherited from System.Object
GetLifetimeService	Inherited from System.MarshalByRefObject
GetType	Inherited from System.Object
InitializeLifetimeService	Inherited from System.MarshalByRefObject
Read	Reads a specified amount from the current XML stream instance and populates the array buffer (Overloaded)
ReadAsync	Returns a Task-based asynchronous version of OracleXmlStream.Read(), which reads a specified amount of data and populates the array buffer (Overloaded)
ReadByte	Inherited from System.IO.Stream
Seek	Sets the position within the current stream and returns the new position within the current stream
SetLength	<i>Not Supported</i>
ToString	Inherited from System.Object
Write	<i>Not Supported</i>
WriteByte	<i>Not Supported</i>
WriteLine	<i>Not Supported</i>

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlStream Class](#)
- [OracleXmlStream Members](#)

Clone

This method creates a copy of an OracleXmlStream object.

Declaration

```
// C#
public object Clone();
```

Return Value

An OracleXmlStream object.

Implements

ICloneable

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

The cloned object has the same property values as that of the object being cloned.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlStream Class](#)
- [OracleXmlStream Members](#)

Close

Overrides Stream

This method closes the current stream and releases any resources associated with it.

Declaration

```
// C#  
public override void Close();
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlStream Class](#)
- [OracleXmlStream Members](#)

Dispose

This public method releases resources allocated by this object.

Declaration

```
// C#  
public void Dispose();
```

Implements

IDisposable

Remarks

The object cannot be reused after being disposed. Although some properties can still be accessed, their values cannot be accountable. Since resources are freed, method calls can lead to exceptions.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlStream Class](#)
- [OracleXmlStream Members](#)

Flush

This method is not supported.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlStream Class](#)
- [OracleXmlStream Members](#)

Read

This method reads a specified amount from the current XML stream instance and populates the array buffer.

Overload List:

- [Read\(byte\[\], int, int\)](#)
This method reads a specified amount of unicode bytes from the current instance, advances the position within the stream, and populates the byte array buffer.
- [Read\(char\[\], int, int\)](#)
This method reads a specified amount of characters from the current instance, advances the position within the stream, and populates the character array buffer.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlStream Class](#)
- [OracleXmlStream Members](#)

Read(byte[], int, int)

Overrides Stream

This method reads a specified amount of unicode bytes from the current instance, advances the position within the stream, and populates the byte array buffer.

Declaration

```
// C#  
public override int Read(byte[] buffer, int offset, int count);
```

Parameters

- *buffer*
The byte array buffer that is populated.
- *offset*
The zero-based offset (in bytes) at which the buffer is populated.
- *count*
The maximum amount of bytes to be read.

Return Value

The number of unicode bytes read into the given byte[] buffer or 0 if the end of the stream has been reached.

Remarks

This method reads a maximum of *count* bytes from the current stream and stores them in buffer beginning at *offset*. The current position within the stream is advanced by the number of bytes read. However, if an exception occurs, the current position within the stream remains unchanged.

The XML data is read starting from the position specified by the *Position* property.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlStream Class](#)
- [OracleXmlStream Members](#)

Read(char[], int, int)

Overrides Stream

This method reads a specified amount of characters from the current instance, advances the position within the stream, and populates the character array buffer.

Declaration

```
// C#  
public override int Read(char[] buffer, int offset, int count);
```

Parameters

- *buffer*
The character array buffer to be populated.
- *offset*
The zero-based offset (in characters) in the buffer at which the buffer is populated.
- *count*
The maximum amount of characters to be read from the stream.

Return Value

The return value indicates the number of characters read from the stream or 0 if the end of the stream has been reached.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

This method requires that the `Position` on the stream instance be zero or an even number.

The XML data is read starting from the position specified by the `Position` property.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlStream Class](#)
- [OracleXmlStream Members](#)

ReadAsync

This method returns a Task-based asynchronous version of `OracleXmlStream.Read()`, which reads a specified amount of data and populates the array buffer.

Overload List:

- [ReadAsync\(byte\[\], int, int, CancellationToken\)](#)
This method returns a Task-based asynchronous version of `OracleXmlStream.Read()`
- [ReadAsync\(char\[\], int, int\)](#)

This method returns a Task-based asynchronous version of `OracleXmlStream.Read()`

- [ReadAsync\(char\[\], int, int, CancellationToken\)](#)

This method returns a Task-based asynchronous version of `OracleXmlStream.Read()`

- [ReadAsync\(byte\[\], int, int\)](#)

This method returns a Task-based asynchronous version of `OracleXmlStream.Read()`

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlStream Class](#)
- [OracleXmlStream Members](#)

ReadAsync(byte[], int, int, CancellationToken)

Declaration

```
// C#  
public override Task<int> ReadAsync(byte[] buffer, int offset, int count,  
CancellationToken cancellationToken);
```

Parameters

- *buffer*
The byte array buffer that is populated.
- *offset*
The zero-based offset (in bytes) at which the buffer is populated.
- *count*
The maximum amount of bytes to be read.
- *cancellationToken*
The input cancellation token which can be used by the application to cancel the task.

Return Value

`Task<int>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Implements

Stream

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:

- The offset or the count parameter is less than 0.
- The offset is greater than or equal to the `buffer.Length`.
- The offset and the count together are greater than the `buffer.Length`.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlStream Class](#)
- [OracleXmlStream Members](#)

ReadAsync(char[], int, int)

This will call into the `ReadAsync` implementation with argument `cancellationToken` passed as `CancellationToken.None`.

Declaration

```
// C#  
public Task<int> ReadAsync(char[] buffer, int offset, int count);
```

Parameters

- *buffer*
The character array buffer to be populated.
- *offset*
The zero-based offset (in characters) at which the buffer is populated.
- *count*
The maximum amount of characters to be read.

Return Value

`Task<int>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Implements

`Stream`

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:

- The offset or the count parameter is less than 0.
- The offset is greater than or equal to the `buffer.Length`.
- The offset and the count together are greater than the `buffer.Length`.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlStream Class](#)
- [OracleXmlStream Members](#)

ReadAsync(char[], int, int, CancellationToken)

Declaration

```
// C#  
public Task<int> ReadAsync(char[] buffer, int offset, int count, CancellationToken  
cancellationToken);
```

Parameters

- *buffer*
The character array buffer to be populated.
- *offset*
The zero-based offset (in characters) at which the buffer is populated.
- *count*
The maximum amount of characters to be read.
- *cancellationToken*
The input cancellation token which can be used by the application to cancel the task.

Return Value

Task<int> object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Implements

Stream

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

ArgumentOutOfRangeException - This exception is thrown if any of the following conditions exist:

- The offset or the count parameter is less than 0.
- The offset is greater than or equal to the buffer.Length.
- The offset and the count together are greater than the buffer.Length.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlStream Class](#)
- [OracleXmlStream Members](#)

ReadAsync(byte[], int, int)

This will call into the ReadAsync implementation with argument `cancellationToken` passed as `CancellationToken.None`.

Declaration

```
// C#  
public Task<int> ReadAsync(byte[] buffer, int offset, int count);
```

Parameters

- *buffer*
The byte array buffer to be populated.
- *offset*
The zero-based offset (in bytes) at which the buffer is populated.
- *count*
The maximum amount of bytes to be read.

Return Value

`Task<int>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Implements

Stream

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:

- The offset or the count parameter is less than 0.
- The offset is greater than or equal to the `buffer.Length`.
- The offset and the count together are greater than the `buffer.Length`.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlStream Class](#)
- [OracleXmlStream Members](#)

Seek

Overrides Stream.

This method sets the position within the current stream and returns the new position within the current stream.

Declaration

```
// C#  
public long Seek(long offset, SeekOrigin origin);
```

Parameters

- *offset*
A byte offset relative to origin.
 - If *offset* is negative, the new position precedes the position specified by *origin* by the number of bytes specified by *offset*.
 - If offset is zero, the new position is the position specified by *origin*.
 - If *offset* is positive, the new position follows the position specified by *origin* by the number of bytes specified by *offset*.
- *origin*
A value of type `SeekOrigin` indicating the reference point used to obtain the new position.

Return Value

The new Position within the current stream.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object

Remarks

Use the `CanSeek` property to determine whether or not the current instance supports seeking. Seeking to any location beyond the length of the stream is supported.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlStream Class](#)
- [OracleXmlStream Members](#)

SetLength

This method is not supported.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlStream Class](#)
- [OracleXmlStream Members](#)

Write

This method is not supported.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlStream Class](#)
- [OracleXmlStream Members](#)

WriteLine

This method is not supported.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlStream Class](#)
- [OracleXmlStream Members](#)

OracleXmlType Class

An `OracleXmlType` object represents an Oracle XMLType instance.

Class Inheritance

System.Object
 System.OracleXmlType

Declaration

```
// C#  
public sealed class OracleXmlType : IDisposable, INullable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

OracleXmlType objects can be used for well-formed XML documents with or without XML schemas or XML fragments.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Members](#)
- [OracleXmlType Constructors](#)
- [OracleXmlType Static Methods](#)
- [OracleXmlType Static Fields](#)
- [OracleXmlType Instance Properties](#)
- [OracleXmlType Instance Methods](#)

OracleXmlType Members

OracleXmlType members are listed in the following tables.

OracleXmlType Constructors

The OracleXmlType constructors are listed in [Table 7-19](#).

Table 7-19 OracleXmlType Constructors

Constructor	Description
OracleXmlType Constructors	Creates an instance of the OracleXmlType class (Overloaded)

OracleXmlType Static Methods

The OracleXmlType static methods are listed in [Table 7-20](#).

Table 7-20 OracleXmlType Static Methods

Methods	Description
Equals	Inherited from System.Object (Overloaded)

OracleXmlType Static Fields

The OracleXmlType static field is listed in [Table 7-21](#).

Table 7-21 OracleXmlType Static Field

Static Field	Description
Null	Represents a null value that can be assigned to an OracleXmlType instance

OracleXmlType Instance Properties

The OracleXmlType instance properties are listed in [Table 7-22](#).

Table 7-22 OracleXmlType Instance Properties

Properties	Description
Connection	Indicates the OracleConnection that is used to retrieve and store XML data in the OracleXmlType
IsEmpty	Indicates whether or not the OracleXmlType is empty
IsFragment	Indicates whether the XML data is a collection of XML elements or a well-formed XML document
IsNull	Indicates whether or not the OracleXmlType is null
IsSchemaBased	Indicates whether or not the XML data represented by the OracleXmlType is based on an XML schema
RootElement	Represents the name of the top-level element of the schema-based XML data contained in the OracleXmlType
Schema	Represents the XML schema of the XML data contained in the OracleXmlType
SchemaUrl	Represents in the database for the XML schema of the XML data contained in the OracleXmlType.

Table 7-22 (Cont.) OracleXmlType Instance Properties

Properties	Description
Value	Returns the XML data starting from the first character in the current instance as a string

OracleXmlType Instance Methods

The OracleXmlType instance methods are listed in [Table 7-23](#).

Table 7-23 OracleXmlType Instance Methods

Methods	Description
Clone	Creates a copy of the OracleXmlType instance
Dispose	Releases the resources allocated by this OracleXmlType object
Equals	Inherited from System.Object
Extract	Extracts a subset from the XML data using the given XPath expression (Overloaded)
GetHashCode	Inherited from System.Object
GetStream	Returns an instance of OracleXmlStream which provides a read-only stream of the XML data stored in this OracleXmlType instance
GetType	Inherited from System.Object
GetXmlDocument	Returns a XmlDocument object containing the XML data stored in this OracleXmlType instance
GetXmlReader	Returns a XmlTextReader object that can be used to manipulate XML data directly using the .NET Framework classes and methods
IsExists	Checks for the existence of a particular set of nodes identified by the given XPath expression in the XMLdata (Overloaded)
ToString	Inherited from System.Object
Transform	Transforms the OracleXmlType into another OracleXmlType instance using the given XSL document (Overloaded)
Update	Updates the XML node or fragment identified by the given XPath expression in the current OracleXmlType instance (Overloaded)
Validate	Validates whether or not the XML data in the OracleXmlType object conforms to the given XML schema.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)

OracleXmlType Constructors

OracleXmlType constructors create instances of the OracleXmlType class.

Overload List:

- [OracleXmlType\(OracleClob\)](#)
This constructor creates an instance of the OracleXmlType class using the XML data contained in an OracleClob object.
- [OracleXmlType\(OracleConnection, string\)](#)
This constructor creates an instance of the OracleXmlType class using the XML data contained in the .NET String.
- [OracleXmlType\(OracleConnection, XmlReader\)](#)
This constructor creates an instance of the OracleXmlType class using the contents of the .NET XmlReader object.
- [OracleXmlType\(OracleConnection, XmlDocument\)](#)
This constructor creates an instance of the OracleXmlType object using the contents of the XML [DOM](#) document in the .NET XmlDocument object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

OracleXmlType(OracleClob)

This constructor creates an instance of the OracleXmlType class using the XML data contained in an OracleClob object.

Declaration

```
// C#  
public OracleXmlType(OracleClob oraClob);
```

Parameters

- *oraClob*
An OracleClob object.

Exceptions

ArgumentNullException - The OracleClob object is null.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

The CLOB data depends on a valid connection object and the new OracleXMLType uses the OracleConnection in the OracleClob object to store data for the current instance.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

OracleXmlType(OracleConnection, string)

This constructor creates an instance of the `OracleXmlType` class using the XML data contained in the .NET String.

Declaration

```
// C#  
public OracleXmlType(OracleConnection con, string xmlData);
```

Parameters

- *con*
An `OracleConnection` object.
- *xmlData*
A string containing the XML data.

Exceptions

`ArgumentNullException` - The `OracleConnection` object is null.

`ArgumentException` - The `xmlData` argument is an empty string.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The new `OracleXmlType` uses the given `OracleConnection` object to store data for the current instance.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

OracleXmlType(OracleConnection, XmlReader)

This constructor creates an instance of the `OracleXmlType` class using the contents of the .NET `XmlReader` object.

Declaration

```
// C#  
public OracleXmlType(OracleConnection con, XmlReader reader);
```

Parameters

- *con*
An OracleConnection object.
- *reader*
An XmlReader object.

Exceptions

ArgumentNullException - The OracleConnection object is null.

ArgumentException - The *reader* argument contains no data.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

The new OracleXMLType uses the given OracleConnection object to store data for the current instance.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

OracleXmlType(OracleConnection, XmlDocument)

This constructor creates an instance of the OracleXmlType object using the contents of the XML [DOM](#) document in the .NET XmlDocument object.

Declaration

```
// C#  
public OracleXmlType(OracleConnection con, XmlDocument domDoc);
```

Parameters

- *con*
An OracleConnection object.
- *domDoc*
An XML document.

Exceptions

ArgumentNullException - The OracleConnection object is null.

ArgumentException - The *domDoc* argument contains no data.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

The new OracleXMLType uses the given OracleConnection object to store data for the current instance.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

OracleXmlType Static Methods

The OracleXmlType static methods are listed in [Table 7-24](#).

Table 7-24 OracleXmlType Static Methods

Methods	Description
Equals	Inherited from System.Object (Overloaded)

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

OracleXmlType Static Fields

The OracleXmlType static field is listed in [Table 7-25](#).

Table 7-25 OracleXmlType Static Field

Static Field	Description
Null	Represents a null value that can be assigned to an OracleXmlType instance

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

Null

This static field represents a null value that can be assigned to an `OracleXmlType` instance.

Declaration

```
// C#
public static readonly OracleXmlType Null;
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

OracleXmlType Instance Properties

The `OracleXmlType` instance properties are listed in [Table 7-26](#).

Table 7-26 OracleXmlType Instance Properties

Properties	Description
Connection	Indicates the <code>OracleConnection</code> that is used to retrieve and store XML data in the <code>OracleXmlType</code>
IsEmpty	Indicates whether or not the <code>OracleXmlType</code> is empty
IsFragment	Indicates whether the XML data is a collection of XML elements or a well-formed XML document
IsNull	Indicates whether or not the <code>OracleXmlType</code> is null
IsSchemaBased	Indicates whether or not the XML data represented by the <code>OracleXmlType</code> is based on an XML schema
Null	Represents a null value that can be assigned to an <code>OracleXmlType</code> instance
RootElement	Represents the name of the top-level element of the schema-based XML data contained in the <code>OracleXmlType</code>
Schema	Represents the XML schema of the XML data contained in the <code>OracleXmlType</code>
SchemaUrl	Represents URL in the database for the XML schema of the XML data contained in the <code>OracleXmlType</code>

Table 7-26 (Cont.) OracleXmlType Instance Properties

Properties	Description
Value	Returns the XML data starting from the first character in the current instance as a string

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

Connection

This property indicates the `OracleConnection` that is used to retrieve and store XML data in the `OracleXmlType`.

Declaration

```
// C#  
public OracleConnection Connection {get;}
```

Property Value

An `OracleConnection` object.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The connection must explicitly be opened by the user before creating or using `OracleXmlType`.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

IsEmpty

This property indicates whether or not the `OracleXmlType` is empty.

Declaration

```
// C#  
public bool IsEmpty {get;}
```

Property Value

Returns true if the OracleXmlType represents an empty XML document. Returns false otherwise.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

IsFragment

This property indicates whether the XML data is a collection of XML elements or a well-formed XML document.

Declaration

```
// C#  
public bool IsFragment {get;}
```

Property Value

Returns true if the XML data contained in the OracleXmlType object is a collection of XML elements with no root element. Returns false otherwise.

Exceptions

ObjectDisposedException - The object is already disposed.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

IsNull

This property indicates whether or not the OracleXmlType is null.

Declaration

```
// C#  
public bool IsNull {get;}
```

Property Value

Returns true if the OracleXmlType represents a null value. Returns false otherwise.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

IsSchemaBased

This property indicates whether or not the XML data represented by the OracleXmlType is based on an XML schema.

Declaration

```
// C#  
public bool IsSchemaBased {get;}
```

Property Value

Returns true if the XML data represented by the OracleXmlType is based on an XML schema. Returns false otherwise.

Exceptions

ObjectDisposedException - The object is already disposed.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

RootElement

This property represents the name of the top-level or root element of the schema-based XML data contained in the `OracleXmlType`.

Declaration

```
// C#  
public string RootElement{get;}
```

Property Value

A string that represents the name of the top-level or root element of the XML data contained in the `OracleXmlType`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

If the `OracleXmlType` instance contains non-schema based XML data, this property returns an empty string.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

Schema

This property represents the XML schema for the XML data contained in the `OracleXmlType`.

Declaration

```
// C#  
public OracleXmlType Schema {get;}
```

Property Value

An `OracleXmlType` instance that represents the XML schema for the XML data contained in the `OracleXmlType`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

If the `OracleXmlType` instance contains non-schema based XML data, this property returns an `OracleXmlType` instance representing an empty XML document.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

SchemaUrl

This property represents the XML schema in the database for the XML schema of the XML data contained in the `OracleXmlType`.

Declaration

```
// C#  
public string SchemaUrl {get;}
```

Property Value

A string that represents the URL in the database for the XML schema of the XML data.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

If the `OracleXmlType` instance contains non-schema based XML data, this property returns an empty string.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

Value

This property returns the XML data starting from the first character in the current instance as a string.

Declaration

```
// C#  
public string Value{get;}
```

Property Value

The entire XML data as a string.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

OracleXmlType Instance Methods

The OracleXmlType instance methods are listed in [Table 7-27](#).

Table 7-27 OracleXmlType Instance Methods

Methods	Description
Clone	Creates a copy of the OracleXmlType instance
Dispose	Releases the resources allocated by this OracleXmlType object
Equals	Inherited from System.Object
Extract	Extracts a subset from the XML data using the given XPath expression (Overloaded)
GetHashCode	Inherited from System.Object
GetStream	Returns an instance of OracleXmlStream which provides a read-only stream of the XML data stored in this OracleXmlType instance
GetType	Inherited from System.Object
GetXmlDocument	Returns a XmlDocument object containing the XML data stored in this OracleXmlType instance
GetXmlReader	Returns a XmlTextReader object that can be used to manipulate XML data directly using the .NET Framework classes and methods
IsExists	Checks for the existence of a particular set of nodes identified by the given XPath expression in the XMLdata (Overloaded)
ToString	Inherited from System.Object
Transform	Transforms the OracleXmlType into another OracleXmlType instance using the given XSL document (Overloaded)
Update	Updates the XML node or fragment identified by the given XPath expression in the current OracleXmlType instance (Overloaded)
Validate	Validates whether or not the XML data in the OracleXmlType object conforms to the given XML schema.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

Clone

This method creates a copy of this OracleXmlType instance.

Declaration

```
// C#  
public object Clone();
```

Implements

ICloneable

Return Value

An OracleXmlType object.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

Dispose

This method releases the resources allocated by this object.

Declaration

```
// C#  
public void Dispose();
```

Implements

IDisposable

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

Extract

This method extracts a subset from the XML data using the given XPath expression.

Overload List:

- [Extract\(string, string\)](#)

This method extracts a subset from the XML data represented by the `OracleXmlType` object using the given XPath expression and a string parameter for namespace resolution.

- [Extract\(string, XmlNamespaceManager\)](#)

This method extracts a subset from the XML data represented by the `OracleXmlType` object, using the given XPath expression and a .NET `XmlNamespaceManager` object for namespace resolution.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

Extract(string, string)

This method extracts a subset from the XML data represented by the `OracleXmlType` object using the given XPath expression and a string parameter for namespace resolution.

Declaration

```
// C#  
public OracleXmlType Extract(string xpathExpr, string nsMap);
```

Parameters

- *xpathExpr*

The XPath expression.

- *nsMap*

The string parameter used for namespace resolution of the XPath expression. *nsMap* has zero or more namespaces separated by spaces. *nsMap* can be null. For example:

```
xmlns:nsi="http://www.company1.com" xmlns:nsz="http://www.company2.com"
```

Return Value

An OracleXmlType object.

Exceptions

ObjectDisposedException - The object is already disposed.

ArgumentNullException - The *xpathExpr* is null or zero-length.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

Extract(string, XmlNamespaceManager)

This public method extracts a subset from the XML data represented by the OracleXmlType object, using the given XPath expression and a .NET XmlNamespaceManager object for namespace resolution.

Declaration

```
// C#  
public OracleXmlType Extract(string xpathExpr, XmlNamespaceManager nsMgr);
```

Parameters

- *xpathExpr*
The XPath expression.
- *nsMgr*
The .NET XmlNamespaceManager object used for namespace resolution of the XPath expression. *nsMgr* can be null.

Return Value

An OracleXmlType.

Exceptions

ObjectDisposedException - The object is already disposed.

ArgumentNullException - The *xpathExpr* is null or zero-length.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

The default namespace is ignored if its value is an empty string.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

GetStream

This public method returns an instance of `OracleXmlStream` which provides a read-only stream of the XML data stored in this `OracleXmlType` instance.

Declaration

```
// C#  
public Stream GetStream();
```

Return Value

A `Stream` object.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

GetXmlDocument

This public method returns a `XmlDocument` object containing the XML data stored in this `OracleXmlType` instance.

Declaration

```
// C#  
public XmlDocument GetXmlDocument();
```

Return Value

An XmlDocument object.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

The XML data in the XmlDocument object is a copy of the XML data in the OracleXmlType instance and modifying it does not automatically modify the XML data in the OracleXmlType instance. The XmlDocument instance returned has the PreserveWhitespace property set to true.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

GetXmlReader

This public method returns a XmlTextReader object that can be used to manipulate XML data directly using the .NET Framework classes and methods.

Declaration

```
// C#  
public XmlTextReader GetXmlReader();
```

Return Value

An XmlTextReader object.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

The XmlTextReader is a read-only, forward-only representation of the XML data stored in the OracleXmlType instance.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

IsExists

IsExists checks for the existence of a particular set of nodes identified by the XPath expression in the XML data.

Overload List:

- [IsExists\(string, string\)](#)
This method checks for the existence of a particular set of nodes identified by the XPath expression in the XML data represented by the current `OracleXmlType` instance using a string parameter for namespace resolution.
- [IsExists\(string, XmlNamespaceManager\)](#)
This method checks for the existence of a particular set of nodes identified by the XPath expression in the XML document represented by the current `OracleXmlType` instance using a .NET `XmlNamespaceManager` object for namespace resolution.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

IsExists(string, string)

This method checks for the existence of a particular set of nodes identified by the XPath expression in the XML data represented by the current `OracleXmlType` instance using a string parameter for namespace resolution.

Declaration

```
// C#  
public bool IsExists(string xpathExpr, string nsMap);
```

Parameters

- *xpathExpr*
The XPath expression.
- *nsMap*
The string parameter used for namespace resolution of the XPath expression. *nsMap* has zero or more namespaces separated by spaces. *nsMap* can be null.

Return Value

Returns true if the required set of nodes exists; otherwise, returns false.

Exceptions

ObjectDisposedException - The object is already disposed.

ArgumentNullException - The *xpathExpr* is null or zero-length.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

The default namespace is ignored if its value is an empty string.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

IsExists(string, XmlNamespaceManager)

This method checks the existence of a particular set of nodes identified by the XPath expression in the XML document represented by the current OracleXmlType instance using a .NET XmlNamespaceManager object for namespace resolution.

Declaration

```
// C#  
public bool IsExists(string xpathExpr, XmlNamespaceManager nsMgr);
```

Parameters

- *xpathExpr*
The XPath expression.
- *nsMgr*
The .NET XmlNamespaceManager object used for namespace resolution of the XPath expression. *nsMgr* can be null.

Return Value

Returns true if the required set of nodes exists; otherwise, returns false.

Exceptions

ObjectDisposedException - The object is already disposed.

ArgumentNullException - The *xpathExpr* is null or zero-length.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

The default namespace is ignored if its value is an empty string.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

Transform

This method transforms the OracleXmlType into another OracleXmlType instance using the given XSL document.

Overload List:

- [Transform\(OracleXmlType, string\)](#)
This method transforms the current OracleXmlType instance into another OracleXmlType instance using the given XSL document (as an OracleXmlType object) and a string of XSLT parameters.
- [Transform\(string, string\)](#)
This public method transforms the current OracleXmlType instance into another OracleXmlType instance using the given XSL document and a string of XSLT parameters.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

Transform(OracleXmlType, string)

This method transforms the current OracleXmlType instance into another OracleXmlType instance using the given XSL document and a string of XSLT parameters.

Declaration

```
// C#  
public OracleXmlType Transform(OracleXmlType xslDoc, string paramMap);
```

Parameters

- *xslDoc*

The XSL document as an `OracleXmlType` object.

- *paramMap*

A string which provides the parameters for the XSL document.

For this release, *paramMap* is ignored.

Return Value

An `OracleXmlType` object containing the transformed XML document.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentNullException` - The *xslDoc* parameter is null.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

Transform(string, string)

This method transforms the current `OracleXmlType` instance into another `OracleXmlType` instance using the given XSL document and a string of XSLT parameters.

Declaration

```
// C#  
public OracleXmlType Transform(string xslDoc, string paramMap);
```

Parameters

- *xslDoc*

The XSL document to be used for XSLT.

- *paramMap*

A string which provides the parameters for the XSL document.

For this release, *paramMap* is ignored.

Return Value

An `OracleXmlType` object containing the transformed XML document.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentNullException` - The *xslDoc* parameter is null.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

Update

This method updates the XML node or fragment identified by the given XPath expression in the current OracleXmlType instance.

Overload List:

- [Update\(string, string, string\)](#)
This method updates the XML nodes identified by the given XPath expression with the given string value and a string parameter for namespace resolution.
- [Update\(string, XmlNamespaceManager, string\)](#)
This method updates the XML nodes identified by the given XPath expression with the given string value and a .NET XmlNamespaceManager object for namespace resolution.
- [Update\(string, string, OracleXmlType\)](#)
This method updates the XML nodes identified by the given XPath expression with the XML data stored in the given OracleXmlType value and a string parameter for namespace resolution.
- [Update\(string, XmlNamespaceManager, OracleXmlType\)](#)
This method updates the XML nodes identified by the given XPath expression with the XML data stored in the given OracleXmlType value and a .NET XmlNamespaceManager object for namespace resolution.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

Update(string, string, string)

This method updates the XML nodes identified by the given XPath expression with the given string value and a string parameter for namespace resolution.

Declaration

```
// C#  
public void Update(string xpathExpr, string nsMap, string value);
```

Parameters

- *xpathExpr*
The XPath expression that identifies the nodes to update.
- *nsMap*
The string parameter used for namespace resolution of the XPath expression. *nsMap* has zero or more namespaces separated by spaces. *nsMap* can be null. For example:
`xmlns:ns1="http://www.company1.com" xmlns:ns2="http://www.company2.com"`
- *value*
The new value as a string.

Exceptions

ObjectDisposedException - The object is already disposed.

ArgumentNullException - The *xpathExpr* is null or zero-length.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

The default namespace is ignored if its value is an empty string.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

Update(string, XmlNamespaceManager, string)

This method updates the XML nodes identified by the given XPath expression with the given string value and a .NET XmlNamespaceManager object for namespace resolution.

Declaration

```
// C#  
public void Update(string xpathExpr, XmlNamespaceManager nsMgr, string  
value);
```

Parameters

- *xpathExpr*
The XPath expression that identifies the nodes to update.

- *nsMgr*
The .NET XmlNamespaceManager object used for namespace resolution of the XPath expression. *nsMgr* can be null.
- *value*
The new value as a string.

Exceptions

ObjectDisposedException - The object is already disposed.

ArgumentNullException - The *xpathExpr* is null or zero-length.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

The default namespace is ignored if its value is an empty string.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

Update(string, string, OracleXmlType)

This method updates the XML nodes identified by the given XPath expression with the XML data stored in the given OracleXmlType value and a string parameter for namespace resolution.

Declaration

```
// C#  
public void Update(string xpathExpr, string nsMap, OracleXmlType value);
```

Parameters

- *xpathExpr*
The XPath expression that identifies the nodes to update.
- *nsMap*
The string parameter used for namespace resolution of the XPath expression. *nsMap* has zero or more namespaces separated by spaces. *nsMap* can be null.
- *value*
The new value as an OracleXmlType object.

Exceptions

ObjectDisposedException - The object is already disposed.

ArgumentNullException - The *xpathExpr* is null or zero-length.

InvalidOperationException - The *OracleConnection* is not open or has been closed during the lifetime of the object.

Remarks

The default namespace is ignored if its value is an empty string.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

Update(string, XmlNamespaceManager, OracleXmlType)

This method updates the XML nodes identified by the given XPath expression with the XML data stored in the given *OracleXmlType* value and a .NET *XmlNamespaceManager* object for namespace resolution.

Declaration

```
// C#  
public void Update(string xpathExpr, XmlNamespaceManager nsMgr, OracleXmlType value);
```

Parameters

- *xpathExpr*
The XPath expression that identifies the nodes to update.
- *nsMgr*
The .NET *XmlNamespaceManager* object used for namespace resolution of the XPath expression. *nsMgr* can be null.
- *value*
The new value as an *OracleXmlType* object.

Exceptions

ObjectDisposedException - The object is already disposed.

ArgumentNullException - The *xpathExpr* is null or zero-length.

InvalidOperationException - The *OracleConnection* is not open or has been closed during the lifetime of the object.

Remarks

The default namespace is ignored if its value is an empty string.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

Validate

This methods validates whether or not the XML data in the `OracleXmlType` object conforms to the given XML schema.

Declaration

```
// C#  
public bool Validate(String schemaUrl);
```

Parameters

- `schemaUrl`
A string representing the [URL](#) in the database of the XML schema.

Return Value

Returns true if the XML data conforms to the XML schema; otherwise, returns false.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentNullException` - The `schemaUrl` argument is null or an empty string.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleXmlType Class](#)
- [OracleXmlType Members](#)

8

Oracle Data Provider for .NET HA Event Classes

This chapter describes the following ODP.NET HA event class and enumerations:

- [OracleHAEventArgs Class](#)
- [OracleHAEventHandler Delegate](#)
- [OracleHAEventSource Enumeration](#)
- [OracleHAEventStatus Enumeration](#)

OracleHAEventArgs Class

The `OracleHAEventArgs` class provides event data for the `OracleConnection.HAEvent` event.

Class Inheritance

`System.Object`

`System.EventArgs`

`Oracle.DataAccess.Client.OracleHAEventArgs`

Declaration

```
// C#  
public sealed class OracleHAEventArgs
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

When any HA event occurs for a service, service member, host, node, or instance that an `OracleConnection` object is set to with "ha_events=true", the `OracleConnection.HAEvent` is

triggered and passes an instance of `OracleHAEventArgs` to all the delegates that have registered with the event.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleHAEventArgs Members](#)
- [OracleHAEventArgs Properties](#)
- ["OracleConnection Class"](#)
- ["HAEvent"](#)

OracleHAEventArgs Members

`OracleHAEventArgs` members are listed in the following table.

OracleHAEventArgs Properties

The `OracleHAEventArgs` properties are listed in [Table 8-2](#).

Table 8-1 OracleHAEventArgs Properties

Name	Description
DatabaseDomainName	Specifies the domain name of the database affected by the HAEvent
DatabaseName	Specifies the database affected by the HAEvent
DrainTimeout	Indicates the number of seconds allowed for resource draining to be completed during a planned outage
HostName	Specifies the host that triggered the event
InstanceName	Specifies the instance that triggered the event
Reason	Specifies the reason for which the HA event was sent by the server
ServiceName	Specifies the service that triggered the event
Source	Specifies the source that triggered the event
Status	Specifies the status of the source that triggered the event
Time	Specifies the time when the event was triggered on the server

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleHAEventArgs Class](#)
- ["HAEvent"](#)

OracleHAEventArgs Properties

The OracleHAEventArgs properties are listed in [Table 8-2](#).

Table 8-2 OracleHAEventArgs Properties

Name	Description
DatabaseDomainName	Specifies the domain name of the database affected by the HAEvent
DatabaseName	Specifies the database affected by the HAEvent
DrainTimeout	Indicates the number of seconds allowed for resource draining to be completed during a planned outage
HostName	Specifies the host that triggered the event
InstanceName	Specifies the instance that triggered the event
Reason	Specifies the reason for which the HA event was sent by the server
ServiceName	Specifies the service that triggered the event
Source	Specifies the source that triggered the event
Status	Specifies the status of the source that triggered the event
Time	Specifies the time when the event was triggered on the server

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleHAEventArgs Class](#)
- [OracleHAEventArgs Members](#)

DatabaseDomainName

This property specifies the domain name of the database that is affected by the HA event.

Declaration

```
// C#  
public string DatabaseDomainName {get;}
```

Property Value

The domain name of the database that is affected by the HA Event.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleHAEventArgs Class](#)
- [OracleHAEventArgs Members](#)
- ["HAEvent"](#)

DatabaseName

This property specifies the database that is affected by the HA event.

Declaration

```
// C#  
public string DatabaseName {get;}
```

Property Value

This property specifies the database name that is affected by the HA event.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleHAEventArgs Class](#)
- [OracleHAEventArgs Members](#)
- ["HAEvent"](#)

DrainTimeout

An integer value indicating the maximum time period in seconds that ODP.NET waits for the service to drain connections from the database service from when the Fast Application Notification SERVICE_DOWN event is received.

Declaration

```
// C#  
public int DrainTimeout {get;}
```

Property Value

The number of seconds allowed for resource draining to be completed during a planned outage

Remarks

When connecting to Oracle Data Guard in Oracle Database 12c Release 2 or later, database administrators can set their own timeout value, `drain_timeout`, to indicate the number of seconds allowed for resource draining to be completed during a planned outage. The `drain_timeout` value populates `OracleHAEventArgs.DrainTimeout.drain_timeout`. `drain_timeout` can be

used by the ODP.NET ServiceRelocationConnectionTimeout to wait for a service to be relocated before attempting new connections.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleHAEventArgs Class](#)
- [OracleHAEventArgs Members](#)
- ["HAEvent"](#)
- [ServiceRelocationConnectionTimeout](#)

HostName

This property specifies the host that triggered the HA event.

Declaration

```
// C#  
public string HostName {get;}
```

Property Value

The host that is affected by the HA Event.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleHAEventArgs Class](#)
- [OracleHAEventArgs Members](#)
- ["HAEvent"](#)

InstanceName

This property specifies the instance that triggered the HA event.

Declaration

```
// C#  
public string InstanceName {get;}
```

Property Value

The instance that is affected by the HA Event.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleHAEventArgs Class](#)
- [OracleHAEventArgs Members](#)
- ["HAEvent"](#)

Reason

This property specifies reason for which the HA event was sent by the server.

Declaration

```
// C#  
public string Reason {get;}
```

Property Value

The reason the HA Event was triggered. Possible values include Data_Guard_Failover, Failure, Dependency, User, Autostart, and Restart.

The value User is indicative of a planned outage. All other values are indicative of an unplanned outage.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleHAEventArgs Class](#)
- [OracleHAEventArgs Members](#)
- ["HAEvent"](#)

ServiceName

This property specifies the service that triggered the HA event.

Declaration

```
// C#  
public string ServiceName {get;}
```

Property Value

The service that is affected by the HA Event.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleHAEventArgs Class](#)
- [OracleHAEventArgs Members](#)
- ["HAEvent"](#)

Source

This property specifies the source that triggered the HA event.

Declaration

```
// C#  
public OracleHAEventSource Source {get;}
```

Property Value

The source that triggered the HA Event.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleHAEventArgs Class](#)
- [OracleHAEventArgs Members](#)
- ["HAEvent"](#)

Status

This property specifies the status of the source that triggered the HA event.

Declaration

```
// C#  
public OracleHAEventStatus Status {get;}
```

Property Value

The status of the source that triggered the HA Event.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleHAEventArgs Class](#)
- [OracleHAEventArgs Members](#)
- ["HAEvent"](#)

Time

This property specifies the time when the HA event was triggered on the server.

Declaration

```
// C#
public DateTime Time {get;}
```

Property Value

The time that the HA Event was triggered.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleHAEventArgs Class](#)
- [OracleHAEventArgs Members](#)
- ["HAEvent"](#)

OracleHAEventHandler Delegate

The OracleHAEventHandler delegate represents the signature of the method that handles the OracleConnection.HAEvent event.

Declaration

```
// C#
public delegate void OracleHAEventHandler(OracleHAEventArgs eventArgs);
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Parameters

- *sender*
The source of the event.
- *EventArgs*
The OracleHAEventArgs object that contains the event data.

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleHAEventArgs Class](#)
- "[HAEvent](#)"

OracleHAEventSource Enumeration

The OracleHAEventSource enumeration indicates the source of the HA event.

[Table 8-3](#) lists all the OracleHAEventSource enumeration values with a description of each enumerated value.

Table 8-3 OracleHAEventSource Enumeration Member Values

Member Name	Description
Service	The source of the HA Event is a service.
ServiceMember	The source of the HA Event is a service member.
Database	The source of the HA Event is a database.
Host	The source of the HA Event is a host.
Instance	The source of the HA Event is an instance.
Node	The source of the HA Event is a node.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleHAEventArgs Class](#)
- "[Source](#)"

OracleHAEventStatus Enumeration

The OracleHAEventStatus enumeration indicates the status of the HA event source.

[Table 8-4](#) lists all the OracleHAEventStatus enumeration values with a description of each enumerated value.

Table 8-4 OracleHAEventStatus Enumeration Values

Member Name	Description
Up	The source of the HA Event is up.
Down	The source of the HA Event is down.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleHAEventArgs Class](#)
- ["Status"](#)

Continuous Query Notification Classes

This chapter describes Oracle Data Provider for .NET Continuous Query Notification Classes, Event Delegates, and Enumerations.

See Also

["Continuous Query Notification Support"](#)

This chapter contains these topics:

- [OracleDependency Class](#)
- [OracleNotificationRequest Class](#)
- [OracleNotificationEventArgs Class](#)
- [OnChangeEventHandler Delegate](#)
- [OracleRowidInfo Enumeration](#)
- [OracleNotificationType Enumeration](#)
- [OracleNotificationSource Enumeration](#)
- [OracleNotificationInfo Enumeration](#)

OracleDependency Class

An `OracleDependency` class represents a dependency between an application and an Oracle database, enabling the application to get notifications whenever the data of interest or the state of the Oracle database changes.

Class Inheritance

`System.Object`

`Oracle.DataAccess.Client.OracleDependency`

Declaration

```
// C#
public sealed class OracleDependency
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although methods do not guarantee thread safety.

 See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDependency Members](#)
- [OracleDependency Constructors](#)
- [OracleDependency Static Fields](#)
- [OracleDependency Static Methods](#)
- [OracleDependency Methods](#)
- [OracleDependency Properties](#)
- [OracleDependency Events](#)

OracleDependency Members

OracleDependency members are listed in the following tables.

OracleDependency Constructors

OracleDependency constructors are listed in [Table 9-1](#).

Table 9-1 OracleDependency Constructors

Constructors	Description
OracleDependency Constructors	Instantiates a new instance of OracleDependency class (Overloaded)

OracleDependency Static Fields

The OracleDependency static fields are listed in [Table 9-2](#).

Table 9-2 OracleDependency Static Fields

Static Field	Description
Address	Indicates the address that the notification listener listens on, for database notifications Supported in ODP.NET Core only
Port	Indicates the port number that the notification listener listens on, for database notifications

OracleDependency Static Methods

OracleDependency static methods are listed in [Table 9-3](#).

Table 9-3 OracleDependency Static Methods

Static Methods	Description
<code>Equals</code>	Inherited from <code>System.Object</code>
GetOracleDependency	Returns an <code>OracleDependency</code> instance based on the specified unique identifier

OracleDependency Properties

OracleDependency properties are listed in [Table 9-4](#).

Table 9-4 OracleDependency Properties

Properties	Description
DataSource	Indicates the data source associated with the <code>OracleDependency</code> instance
HasChanges	Indicates whether or not there is any change in the database associated with this dependency
Id	Represents the unique identifier for the <code>OracleDependency</code> instance
IsEnabled	Specifies whether or not the dependency is enabled between the application and the database
QueryBasedNotification	Specifies whether the change notification registration is object-based or query-based
RegisteredQueryIDs	Provides a list of <code>CHANGE_NOTIFICATION_QUERY_IDS</code>
RegisteredResources	Indicates the database resources that are registered in the notification registration
RowidInfo	Specifies whether or not ROWID information is part of change notification events fired whenever data changes on the database
UserName	Indicates the database user name associated with the <code>OracleDependency</code> instance

OracleDependency Methods

OracleDependency methods are listed in [Table 9-5](#).

Table 9-5 OracleDependency Methods

Methods	Description
AddCommandDependency	Binds the OracleDependency instance to the specified OracleCommand instance
Equals	Inherited from System.Object
GetHashCode	Inherited from System.Object
GetType	Inherited from System.Object
RemoveRegistration	Removes the specified dependency between the application and the database
ToString	Inherited from System.Object

OracleDependency Events

The OracleDependency event is listed in [Table 9-6](#).

Table 9-6 OracleDependency Events

Event	Description
OnChange	An event that is sent when a database notification associated with the dependency is received from the database

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDependency Class](#)

OracleDependency Constructors

OracleDependency constructors create instances of the OracleDependency class.

Overload List:

- [OracleDependency \(\)](#)
This constructor creates an instance of the OracleDependency class.
- [OracleDependency\(OracleCommand\)](#)
This constructor creates an instance of the OracleDependency class and binds it to the specified OracleCommand instance.
- [OracleDependency\(OracleCommand, bool, int, bool\)](#)
This constructor creates an instance of the OracleDependency class and binds it to the specified OracleCommand instance, specifying whether or not a notification is to be removed upon notification, the timeout value of the notification registration, and the persistence of the notification.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDependency Class](#)
- [OracleDependency Members](#)

OracleDependency ()

This constructor creates an instance of the `OracleDependency` class.

Declaration

```
// C#  
public OracleDependency ( )
```

Remarks

Using this constructor does not bind any `OracleCommand` to the newly constructed `OracleDependency`. Use the `AddCommandDependency` method to do so.

① Note

The dependency between the application and the database is not established when the `OracleDependency` instance is created. The dependency is established when the command that is associated with this dependency is executed.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDependency Class](#)
- [OracleDependency Members](#)

OracleDependency(OracleCommand)

This constructor creates an instance of the `OracleDependency` class and binds it to an `OracleCommand` instance.

Declaration

```
// C#  
public OracleDependency (OracleCommand cmd)
```

Parameters

- *cmd*

The command that the `OracleDependency` object binds to.

Exceptions

ArgumentNullException - The *cmd* parameter is null.

InvalidOperationException - The specified OracleCommand instance already contains a notification request.

Remarks

When this OracleDependency constructor binds the OracleCommand instance to an OracleDependency instance, it causes the creation of an OracleNotificationRequest instance and then sets that OracleNotificationRequest instance to the OracleCommand.Notification property.

The Continuous Query Notification is registered with the database, when the command is executed. Any of the command execution methods (for example, ExecuteNonQuery, ExecuteReader, and so on) will register the notification request. An OracleDependency may be bound to more than one OracleCommand. When one of these OracleCommand object statements is executed, the statement is registered with the associated OracleCommand. Although the registration happens on each OracleCommand separately, one OracleDependency can be responsible for detecting and sending notifications that occur for all OracleCommand objects that the OracleDependency is associated with. The OnChangeEventArgs that is passed to the application for the OnChange event provides information on what has changed in the database.

The OracleNotificationRequest instance that is created by this constructor has the following default property values:

- IsNotifiedOnce is set to the value True.
- Timeout is set to 50,000 seconds.
- IsPersistent is set to the value False, that is, the invalidation message is not persistent, but is stored in an in-memory queue before delivery.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDependency Class](#)
- [OracleDependency Members](#)

OracleDependency(OracleCommand, bool, int, bool)

This constructor creates an instance of the OracleDependency class and binds it to the specified OracleCommand instance, while specifying whether or not a registration is to be removed upon notification, the timeout value of the notification registration, and the persistence of the notification.

Declaration

```
// C#
public OracleDependency (OracleCommand cmd, bool isNotifiedOnce, long timeout,
    bool isPersistent)
```

Parameters

- *cmd*
The command associated with the Continuous Query Notification request.
- *isNotifiedOnce*
An indicator that specifies whether or not the registration is removed automatically once the notification occurs.
- *timeout*
The amount of time, in seconds, that the registration stays active. When *timeout* is set to 0, the registration never expires. The valid values for *timeout* are between 0 and 4294967295.
- *isPersistent*
Indicates whether or not the invalidation message should be queued persistently in the database before delivery. If the *isPersistent* parameter is set to True, the message is queued persistently in the database and cannot be lost upon database failures or shutdowns. If the *isPersistent* property is set to False, the message is stored in an in-memory queue before delivery and might be lost.

Database performance is faster if the message is stored in an in-memory queue rather than in the database queue.

Exceptions

ArgumentNullException - The *cmd* parameter is null.

ArgumentOutOfRangeException - The specified *timeout* is invalid.

InvalidOperationException - The specified *OracleCommand* instance already contains a notification request.

Remarks

When this *OracleDependency* constructor binds the *OracleCommand* instance to an *OracleDependency* instance, it causes the creation of an *OracleNotificationRequest* instance and then sets that *OracleNotificationRequest* instance to the *OracleCommand.Notification* property.

The Continuous Query Notification is registered with the database, when the command is executed. Any of the command execution methods (for example, *ExecuteNonQuery*, *ExecuteReader*, and so on) will register the notification request. An *OracleDependency* may be bound to more than one *OracleCommand*. When one of these *OracleCommand* object statements is executed, the statement is registered with the associated *OracleCommand*. Although the registration happens on each *OracleCommand* separately, one *OracleDependency* can be responsible for detecting and sending notifications that occur for all *OracleCommand* objects that the *OracleDependency* is associated with. The *OnChangeEventArgs* that is passed to the application for the *OnChange* event provides information on what has changed in the database.

The *OracleNotificationRequest* instance that is created by this constructor has the following default property values:

- *IsNotifiedOnce* is set to the specified value.
- *Timeout* is set to the specified value.
- *IsPersistent* is set to the specified value.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDependency Class](#)
- [OracleDependency Members](#)

OracleDependency Static Fields

The OracleDependency static fields are listed in [Table 9-7](#).

Table 9-7 OracleDependency Static Fields

Static Field	Description
Address	Indicates the address that the notification listener listens on, for database notifications Supported in ODP.NET Core only
Port	Indicates the port number that the notification listener listens on, for database notifications

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDependency Class](#)
- [OracleDependency Members](#)

Address

This static field indicates the address that the notification listener listens on, for database notifications.

Declaration

```
// C#  
public static string Address{get; set}
```

Property Value

A string value that represents the address that listens for the database notifications.

If the address is not set, then `OracleConfiguration.DBNotificationAddress` will be used if set. If `OracleDependency.Address` is explicitly set to `string.Empty` or `null`, then the hostname will be used.

Remarks

This property allows specifying a particular IPv4 or IPv6 address to use, such as the case with a machine that contains multiple network cards. The address can be set to any valid

hostname, IPv4, or IPv6 address, which is associated with the machine on which the ODP.NET application is running on, which will be listening for notifications.

The address specified by the `OracleDependency.Address` static field is used by the notification listener that runs within the same application domain as ODP.NET. This address receives Continuous Query Notifications from the database. One notification listener is capable of listening to all Continuous Query Notifications and therefore, only one notification listener is created for each application domain.

The notification listener is created when a command associated with an `OracleDependency` object is executed for the first time during the application domain lifetime. The address specified for the `OracleDependency.Address` static field is used by the listener for its lifetime. The `OracleDependency.Address` static field can be changed after the creation of the notification listener, but doing so does not affect the actual address that the notification listener listens on.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDependency Class](#)
- [OracleDependency Members](#)

Port

This static field indicates the port number that the notification listener listens on, for database notifications.

Declaration

```
// C#  
public static int Port{get; set}
```

Property Value

An `int` value that represents the number of the port that listens for the database notifications. If the port number is set to -1, a random port number is assigned for the listener when the listener is started. Otherwise, the specified port number is used to start the listener.

Exceptions

`ArgumentOutOfRangeException` - The port number is set to a negative value.

`InvalidOperationException` - The port number is being changed after the listener has started.

Remarks

The port number specified by the `OracleDependency.Port` static field is used by the notification listener that runs within the same application domain as ODP.NET. This port number receives Continuous Query Notifications from the database. One notification listener is capable of listening to all Continuous Query Notifications and therefore, only one notification listener is created for each application domain.

The notification listener is created when a command associated with an `OracleDependency` object is executed for the first time during the application domain lifetime. The port number specified for the `OracleDependency.Port` static field is used by the listener for its lifetime. The

OracleDependency.Port static field can be changed after the creation of the notification listener, but doing so does not affect the actual port number that the notification listener listens on.

📘 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDependency Class](#)
- [OracleDependency Members](#)

OracleDependency Static Methods

OracleDependency static methods are listed in [Table 9-8](#).

Table 9-8 OracleDependency Static Methods

Static Methods	Description
Equals	Inherited from System.Object
GetOracleDependency	Returns an OracleDependency instance based on the specified unique identifier

📘 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDependency Class](#)
- [OracleDependency Members](#)

GetOracleDependency

This static method returns an OracleDependency instance based on the specified unique identifier.

Declaration

```
// C#  
public static OracleDependency GetOracleDependency(string guid)
```

Parameters

- *guid*
The string representation of the unique identifier of an OracleDependency instance.

Exceptions

ArgumentException - The specified unique identifier cannot locate an OracleDependency instance.

Return Value

An OracleDependency instance that has the specified *guid* parameter.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDependency Class](#)
- [OracleDependency Members](#)

OracleDependency Properties

OracleDependency properties are listed in [Table 9-9](#).

Table 9-9 OracleDependency Properties

Properties	Description
DataSource	Indicates the data source associated with the OracleDependency instance
HasChanges	Indicates whether or not there is any change in the database associated with this dependency
Id	Represents the unique identifier for the OracleDependency instance
IsEnabled	Specifies whether or not the dependency is enabled between the application and the database
QueryBasedNotification	Specifies whether the change notification registration is object-based or query-based
RegisteredQueryIDs	Provides a list of CHANGE_NOTIFICATION_QUERY_IDs
RegisteredResources	Indicates the database resources that are registered in the notification registration
RowidInfo	Specifies whether or not ROWID information is part of change notification events fired whenever data changes on the database
UserName	Indicates the database user name associated with the OracleDependency instance

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleDependency Class](#)
- [OracleDependency Members](#)

DataSource

This property indicates the data source associated with the OracleDependency instance.

Declaration

```
// C#  
public string DataSource{get;}
```

Property Value

A string that indicates the data source associated with the `OracleDependency` instance.

Remarks

The `DataSource` property is populated with the data source once the `OracleCommand` associated with the `OracleDependency` executes and registers for the notification successfully.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDependency Class](#)
- [OracleDependency Members](#)

HasChanges

This property indicates whether or not there is any change in the database associated with this dependency.

Declaration

```
// C#  
public bool HasChanges{get;}
```

Property Value

A `bool` value that returns `True` if the database has detected changes that are associated with this dependency; otherwise, returns `False`.

Remarks

As an alternative to using the `OnChange` event, applications can check the `HasChanges` property to determine if there are any changes in the database associated with this dependency.

Once the `HasChanges` property is accessed, its value is reset to `False` so that the next notification can then be acknowledged.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDependency Class](#)
- [OracleDependency Members](#)

Id

This property represents the unique identifier for the OracleDependency instance.

Declaration

```
// C#  
public string Id{get;}
```

Property Value

A string that represents the unique identifier that was generated for the OracleDependency instance when it was created.

Remarks

This property is set when the OracleDependency instance is created.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDependency Class](#)
- [OracleDependency Members](#)

IsEnabled

This property specifies whether or not the dependency is enabled between the application and the database.

Declaration

```
// C#  
public bool IsEnabled {get;}
```

Property Value

A bool value that specifies whether or not dependency is enabled between the application and the database.

Remarks

The dependency between the application and the database is not established when the OracleDependency instance is created. The dependency is established when the command that is associated with this dependency is executed, at which time the notification request is registered with the database. The dependency ends when the notification registration is removed from the database or when it times out.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDependency Class](#)
- [OracleDependency Members](#)

QueryBasedNotification

This instance property specifies whether the change notification registration is object-based or query-based.

Declaration

```
// C#  
public bool QueryBasedNotification{get; set;}
```

Property Value

Specifies whether the change notification registration is object-based or not.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

This property value will be ignored if it is set after the command execution that registers the command for change notification.

By default, this property is true.

ODP.NET developers can register their queries on the row level, not just the object level, beginning with Oracle Data Provider for .NET release 11.1 and Oracle Database 11g release 1 (11.1). The application only receives notification when the selected row or rows change. Query-based notifications provide developers more granularity for using client-side cached data, as they can be more specific about what changes the application needs to be notified of.

`OracleNotificationType` enumeration is set to `Query` for query-based notifications.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDependency Class](#)
- [OracleDependency Members](#)

RegisteredQueryIDs

This instance property provides a list of `CHANGE_NOTIFICATION_QUERY_IDS`.

Declaration

```
// C#  
public ArrayList RegisteredQueryIDs {get;}
```

Property Value

This property is an ArrayList of CHANGE_NOTIFICATION_QUERY_IDs.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

This property provides a list of CHANGE_NOTIFICATION_QUERY_IDs that uniquely identify the query that has been registered for change notification. The notification returned from the database will also contain these IDs, allowing applications to determine the query that the notifications are for.

The QueryId at index *n* in RegisteredQueryIDs is for the statement at index *n* the RegisteredResources at index *n*.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDependency Class](#)
- [OracleDependency Members](#)

RegisteredResources

This property indicates the database resources that are registered in the notification registration.

Declaration

```
// C#  
public ArrayList RegisteredResources{get;}
```

Property Value

The registered resources in the notification registration.

Remarks

The ArrayList contains all the command statement or statements that are registered for notification through this OracleDependency object. It is appropriately updated when the Continuous Query Notification is registered by a command execution.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDependency Class](#)
- [OracleDependency Members](#)

RowidInfo

This property specifies whether or not ROWID information is part of change notification events fired whenever data changes on the database.

Declaration

```
// C#  
public OracleRowidInfo RowidInfo {get; set;;}
```

Property Value

An `OracleRowidInfo` enumeration type that determines the inclusion of ROWID in the change notification event.

Remarks

There are three `OracleRowidInfo` enumeration types that are valid for this property:

- `Default` includes ROWID information in the change notification event only if `OracleCommand.AddRowid` property is set to true or if ROWID is in the select list of the query that is registered for change notification.
- `Include` includes ROWID information regardless of whether or not ROWID is in the select-list for the query.
- `Exclude` excludes ROWID information regardless of whether or not ROWID is in the select-list.

For change notification registrations that involve stored procedure executions, change notification events related to the REF CURSOR contain ROWID information only if `RowidInfo` property is set to `OracleRowidInfo.Include`.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDependency Class](#)
- [OracleDependency Members](#)
- ["OracleRowidInfo Enumeration"](#)

UserName

This property indicates the database user name associated with the `OracleDependency` instance.

Declaration

```
// C#  
public string UserName{get;}
```

Property Value

A string that indicates the database user name associated with the `OracleDependency` instance. This database user registers the Continuous Query Notification request with the database.

Remarks

The `UserName` property is populated with the user name once the `OracleCommand` associated with the `OracleDependency` executes and registers for the notification successfully. Only the database user who creates the notification registration, or the database system administrator, can remove the registration.

The user specified by this property must have the `CHANGE NOTIFICATION` privilege to register successfully for the Continuous Query Notification with the database.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDependency Class](#)
- [OracleDependency Members](#)

OracleDependency Methods

`OracleDependency` methods are listed in [Table 9-10](#).

Table 9-10 OracleDependency Methods

Methods	Description
AddCommandDependency	Binds the <code>OracleDependency</code> instance to the specified <code>OracleCommand</code> instance
<code>Equals</code>	Inherited from <code>System.Object</code>
<code>GetHashCode</code>	Inherited from <code>System.Object</code>
<code>GetType</code>	Inherited from <code>System.Object</code>
RemoveRegistration	Removes the specified dependency between the application and the database
<code>ToString</code>	Inherited from <code>System.Object</code>

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDependency Class](#)
- [OracleDependency Members](#)

AddCommandDependency

This instance method binds the `OracleDependency` instance to the specified `OracleCommand` instance.

Declaration

```
// C#  
Public void AddCommandDependency (OracleCommand cmd);
```

Parameters

- *cmd*

The command that is to be bound to the `OracleDependency` object.

Exceptions

`ArgumentNullException` - The *cmd* parameter is null.

`InvalidOperationException` - The specified `OracleCommand` instance already contains a notification request.

Remarks

An `OracleDependency` instance can bind to multiple `OracleCommand` instances.

While it binds an existing `OracleDependency` instance to an `OracleCommand` instance, the `AddCommandDependency` method creates an `OracleNotificationRequest` instance, and sets it to the specified `OracleCommand.Notification` property.

When this method creates an `OracleNotificationRequest` instance, the following `OracleNotificationRequest` properties are set:

- `IsNotifiedOnce` is set to the value `True`.
- `Timeout` is set to 50,000 seconds.
- `IsPersistent` is set to the value `False`, indicating that the invalidation message is stored in an in-memory queue before delivery.

With this method, multiple commands can be associated with a single Continuous Query Notification registration request. Furthermore, the `OracleNotificationRequest` attribute values assigned to the `OracleCommand` can be changed once the association between the `OracleCommand` and the `OracleDependency` is established.

However, when multiple `OracleCommand` objects are associated with a single `OracleDependency` object, the `OracleNotificationRequest` attributes (`Timeout`, `IsPersistent`, and `IsNotifiedOnce`) of the first executed `OracleCommand` object are used for registration, the attributes associated with subsequent `OracleCommand` executions will be ignored.

Furthermore, once a command associated with an `OracleDependency` is executed and registered, all other subsequent command executions and registration associated with the same `OracleDependency` must use a connection with the same "User Id" and "Data Source" connection string attribute value settings.

Otherwise, an exception will be thrown.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDependency Class](#)
- [OracleDependency Members](#)
- ["OracleDependency\(OracleCommand\)"](#) for `OracleNotificationRequest` property value

RemoveRegistration

This instance method removes the specified dependency between the application and the database. Once the registration of the dependency is removed from the database, the `OracleDependency` is no longer able to detect any changes that the database undergoes.

Declaration

```
// C#  
public void RemoveRegistration(OracleConnection con)
```

Parameters

- *con*
The connection associated with the `OracleDependency` instance.

Exceptions

`InvalidOperationException` - The associated connection is not open.

Remarks

The notification registration associated with the `OracleDependency` instance is removed from the database.

The `OracleConnection` parameter must be in an *opened state*. This instance method does not open the connection implicitly for the application.

An exception is thrown if the dependency is not valid.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDependency Class](#)
- [OracleDependency Members](#)

OracleDependency Events

The OracleDependency event is listed in [Table 9-11](#).

Table 9-11 OracleDependency Event

Event	Description
OnChange	An event that is sent when a database notification associated with the dependency is received from the database

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDependency Class](#)
- [OracleDependency Members](#)

OnChange

The OnChange event is sent when a database notification associated with the dependency is received from the database. The information related to the notification is stored in the OracleChangeNotificationEventArgs class.

Declaration

```
// C#
public event OnChangeEventHandler OnChange;
```

Remarks

The OnChange event occurs if any result set associated with the dependency changes. For objects that are part of a Transaction, notifications will be received for each modified object. This event also occurs for other actions related to database or registration status, such as database shutdowns and startups, or registration timeouts.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDependency Class](#)
- [OracleDependency Members](#)

OracleNotificationRequest Class

An OracleNotificationRequest class represents a notification request to be subscribed in the database. It contains information about the request and the characteristics of the notification.

Using the `OracleNotificationRequest` class, Oracle Data Provider for .NET can create the notification registration in the database.

Class Inheritance

System.Object

Oracle.DataAccess.Client.OracleNotificationRequest

Declaration

```
// C#
public sealed class OracleNotificationRequest
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although methods do not guarantee thread safety.

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleNotificationRequest Members](#)
- [OracleNotificationRequest Static Methods](#)
- [OracleNotificationRequest Properties](#)
- [OracleNotificationRequest Methods](#)

OracleNotificationRequest Members

`OracleNotificationRequest` members are listed in the following tables.

OracleNotificationRequest Static Method

The `OracleNotificationRequest` static method is listed in [Table 9-12](#).

Table 9-12 OracleNotificationRequest Static Method

Static Method	Description
Equals	Inherited from System.Object

OracleNotificationRequest Properties

OracleNotificationRequest properties are listed in [Table 9-13](#).

Table 9-13 OracleNotificationRequest Properties

Properties	Description
IsNotifiedOnce	Indicates whether or not the registration is to be removed upon notification
IsPersistent	Indicates whether or not the notification message should be queued persistently in the database before delivery
Timeout	Specifies the time that the registration remains alive
GroupingNotificationEnabled	Specifies whether grouping notification is enabled or not
GroupingType	Specifies the type of grouping notification
GroupingInterval	Specifies the interval between grouping notifications, in seconds

OracleNotificationRequest Methods

OracleNotificationRequest methods are listed in [Table 9-14](#).

Table 9-14 OracleNotificationRequest Methods

Methods	Description
Equals	Inherited from System.Object
GetHashCode	Inherited from System.Object
GetType	Inherited from System.Object
ToString	Inherited from System.Object

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleNotificationRequest Members](#)
- [OracleNotificationRequest Class](#)

OracleNotificationRequest Static Methods

The OracleNotificationRequest static method is listed in [Table 9-15](#).

Table 9-15 OracleNotificationRequest Static Method

Static Method	Description
<code>Equals</code>	Inherited from <code>System.Object</code>

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleNotificationRequest Members](#)
- [OracleNotificationRequest Class](#)

OracleNotificationRequest Properties

The `OracleNotificationRequest` properties are listed in [Table 9-16](#).

Table 9-16 OracleNotificationRequest Properties

Properties	Description
IsNotifiedOnce	Indicates whether or not the registration is to be removed upon notification
IsPersistent	Indicates whether or not the notification message should be queued persistently in the database before delivery
Timeout	Specifies the time that the registration remains alive
GroupingNotificationEnabled	Specifies whether grouping notification is enabled or not
GroupingType	Specifies the type of grouping notification
GroupingInterval	Specifies the interval between grouping notifications, in seconds

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleNotificationRequest Members](#)
- [OracleNotificationRequest Class](#)

IsNotifiedOnce

This property indicates whether or not the registration is to be removed upon notification.

Declaration

```
// C#  
public bool IsNotifiedOnce { get; set; }
```

Property Value

A bool value that indicates whether or not the registration is to be removed upon notification.

Remarks

The default value is false for AQ. This is different from change notification where the default value is true.

Modifying this property after the completion of a successful registration has no effect.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleNotificationRequest Members](#)
- [OracleNotificationRequest Class](#)

IsPersistent

This property indicates whether or not the notification message should be queued persistently in the database until delivery.

Declaration

```
// C#  
public bool IsPersistent{get; set;}
```

Property Value

A bool value that indicates whether or not the notifications should be stored persistently in the database until delivery.

When the IsPersistent property is set to True, the message is queued persistently in the database and cannot be lost upon database failures or shutdowns. When the IsPersistent property is set to False, the message is stored in an in-memory queue before delivery and could be lost.

This property does not apply to NotificationRegistration which is always persistent.

This property only applies to the notification message after it has been sent.

Remarks

The default value is false.

The database performs faster if the message is stored in an in-memory queue rather than a database queue.

Modifying this property after the completion of a successful registration has no effect.

This property is ignored for grouping notifications.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleNotificationRequest Members](#)
- [OracleNotificationRequest Class](#)

Timeout

This property specifies the time, in seconds, that the registration remains alive.

Declaration

```
// C#  
public long Timeout{get; set}
```

Property Value

A long value that specifies the time, in seconds, that the registration remains alive. The valid values for the Timeout property are between 0 and 4294967295.

Exceptions

ArgumentOutOfRangeException - The specified Timeout is invalid.

Remarks

The default value is 0 (infinite) for AQ and 50000 for change notification. If the Timeout property is set to 0, then the registration does not expire.

If the registration is removed because the Timeout value has been reached, then the database sends a notification indicating the expiration.

Modifying this property after the completion of a successful registration has no effect.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleNotificationRequest Members](#)
- [OracleNotificationRequest Class](#)

GroupingNotificationEnabled

This property specifies whether to group notifications together by time is enabled or not.

Declaration

```
// C#  
public bool GroupingNotificationEnabled {get; set}
```

Property Value

A true value indicates that grouping notification is enabled. A false value indicates that grouping notification is disabled.

Remarks

The default value is false.

Modifying this property after the completion of a successful registration has no effect.

If enabled, then ODP.NET will group notifications together that occur during the same time period. The time period is defined by `OracleNotificationRequest.GroupingInterval`.

If disabled, then notifications will be generated immediately after the event that triggers them.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleNotificationRequest Members](#)
- [OracleNotificationRequest Class](#)

GroupingType

This property specifies the type of grouping notification, which can summarize all notifications during the time period or only the last notification.

Declaration

```
// C#  
public OracleAQNotificationGroupingType GroupingType {get; set;}
```

Property Value

An `OracleAQNotificationGroupingType` enum value. The possible enum values are `OracleAQNotificationGroupingType.Summary` and `OracleAQNotificationGroupingType.Last`.

Remarks

The default value is `OracleAQNotificationGroupingType.Summary`.

Modifying this property after the completion of a successful registration has no effect.

`OracleAQNotificationGroupingType.Summary` : All notifications in the group are summarized into a single notification.

① Note

The single notification does not include ROWIDs, even if ROWID information was requested to be returned.

OracleAQNotificationGroupingType.Last : Only the last notification in the group is published. The earlier ones discarded.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleNotificationRequest Members](#)
- [OracleNotificationRequest Class](#)

GroupingInterval

This property specifies the interval of grouping notification in seconds. The group notifications are delivered at intervals specified by this property.

Declaration

```
// C#  
public int GroupingInterval {get; set}
```

Property Value

An integer specifying the grouping interval in seconds.

Remarks

If this value is 900, then notifications generated in the same 15 minute interval are grouped together into a single notification. The default value is 600 seconds.

The range of GroupingInterval is from 1 to Int32.MaxValue.

Modifying this property after the completion of a successful registration has no effect.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleNotificationRequest Members](#)
- [OracleNotificationRequest Class](#)

OracleNotificationRequest Methods

OracleNotificationRequest methods are listed in [Table 9-17](#).

Table 9-17 OracleNotificationRequest Methods

Methods	Description
Equals	Inherited from System.Object
GetHashCode	Inherited from System.Object

Table 9-17 (Cont.) OracleNotificationRequest Methods

Methods	Description
GetType	Inherited from System.Object
ToString	Inherited from System.Object

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleNotificationRequest Members](#)
- [OracleNotificationRequest Class](#)

OracleNotificationEventArgs Class

The OracleNotificationEventArgs class provides event data for a notification.

Class Inheritance

System.Object
 System.EventArgs
 Oracle.DataAccess.Client.OracleNotificationEventArgs

Declaration

```
// C#
public sealed class OracleNotificationEventArgs
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although methods do not guarantee thread safety.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleNotificationEventArgs Members](#)
- [OracleNotificationEventArgs Static Fields](#)
- [OracleNotificationEventArgs Static Methods](#)
- [OracleNotificationEventArgs Properties](#)
- [OracleNotificationEventArgs Methods](#)

OracleNotificationEventArgs Members

OracleNotificationEventArgs members are listed in the following tables.

OracleNotificationEventArgs Static Fields

The OracleNotificationEventArgs static field is listed in [Table 9-18](#).

Table 9-18 OracleNotificationEventArgs Static Field

Static Field	Description
Empty	Inherited from System.EventArgs

OracleNotificationEventArgs Static Methods

The OracleNotificationEventArgs static method is listed in [Table 9-19](#).

Table 9-19 OracleNotificationEventArgs Static Method

Static Method	Description
Equals	Inherited from System.Object

OracleNotificationEventArgs Properties

OracleNotificationEventArgs properties are listed in [Table 9-20](#).

Table 9-20 OracleNotificationEventArgs Properties

Properties	Description
Details	Contains detailed information about the current notification
Info	Indicates the database events for the notification
ResourceNames	Indicates the database resources related to the current notification
Source	Returns the database event source for the notification
Type	Returns the database event type for the notification

OracleNotificationEventArgs Methods

OracleNotificationEventArgs methods are listed in [Table 9-21](#).

Table 9-21 OracleNotificationEventArgs Methods

Methods	Description
Equals	Inherited from System.Object
GetHashCode	Inherited from System.Object
GetType	Inherited from System.Object
ToString	Inherited from System.Object

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleNotificationEventArgs Class](#)

OracleNotificationEventArgs Static Fields

The OracleNotificationEventArgs static field is listed in [Table 9-22](#).

Table 9-22 OracleNotificationEventArgs Static Field

Static Field	Description
Empty	Inherited from System.EventArgs

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleNotificationEventArgs Class](#)
- [OracleNotificationEventArgs Members](#)

OracleNotificationEventArgs Static Methods

The OracleNotificationEventArgs static method is listed in [Table 9-23](#).

Table 9-23 OracleNotificationEventArgs Static Method

Static Method	Description
Equals	Inherited from System.Object

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleNotificationEventArgs Class](#)
- [OracleNotificationEventArgs Members](#)

OracleNotificationEventArgs Properties

OracleNotificationEventArgs properties are listed in [Table 9-24](#).

Table 9-24 OracleNotificationEventArgs Properties

Properties	Description
Details	Contains detailed information about the current notification
Info	Indicates the database events for the notification
ResourceNames	Indicates the database resources related to the current notification
Source	Returns the database event source for the notification
Type	Returns the database event type for the notification

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleNotificationEventArgs Class](#)
- [OracleNotificationEventArgs Members](#)

Details

This property contains detailed information about the current notification.

Declaration

```
// C#
Public DataTable Details{get;}
```

Property Value

A DataTable instance that contains detailed information about the current notification.

Remarks

The returned DataTable object contains column data about the current notification in order as shown in [Table 9-25](#).

Table 9-25 DataTable Object Column Data

Name	Type	Description
ResourceName	System.String	The resource name of the invalidated object in the format <Schema_name>.<object_name>
Info	OracleNotificationInfo	The information about the database event that occurs on a resource
Rowid	System.String	The rowid for the invalidated table row
QueryId	Int32	The CHANGE_NOTIFICATION_QUERY_ID

The QueryId column contains the CHANGE_NOTIFICATION_QUERY_ID that corresponds to the pseudo-column that may have been retrieved by a SELECT statement at the time of the query-based notification. Also, the OracleDependency object maintains all the CHANGE_NOTIFICATION_QUERY_IDS that are registered with it.

For Continuous Query Notification:

- The Details property indicates changes for each invalidated object in the notification in the data table.
- If ROWID information is requested, then the ROWID information is populated into the Rowid column. However, if many rows are modified in a table, then the whole table is invalidated, and ROWID information is not provided. Therefore, the Rowid column contains all Null values.
- If the database event is related to a DDL change of the table or a table drop, then the Rowid column is set to Null.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleNotificationEventArgs Class](#)
- [OracleNotificationEventArgs Members](#)

Info

This property indicates the database events for the notification.

Declaration

```
// C#
public OracleNotificationInfo Info{get;}
```

Property Value

An OracleNotificationInfo value that indicates the database event for the notification.

Remarks

The OracleNotificationInfo value is an enumeration type. If several events are received from the invalidation message, the Info property is set to one of the OracleNotificationInfo enumeration values associated with the database events. For example, if a table has been

altered and a new row has been inserted into another table, the Info property is set to either OracleNotificationInfo.Altered or OracleNotificationInfo.Insert.

To obtain more detailed information from the invalidation message, use the Details and the ResourceNames properties.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleNotificationEventArgs Class](#)
- [OracleNotificationEventArgs Members](#)
- ["Details"](#)
- ["ResourceNames"](#)
- ["OracleNotificationInfo Enumeration"](#)

ResourceNames

This property indicates the database resources related to the current notification.

Declaration

```
// C#  
public string[] ResourceNames{get;}
```

Property Value

A string array that indicates the database resources related to the current notification.

Remarks

For Continuous Query Notification, the ResourceNames property contains information about the invalidated object names in the format <schema_name>.<object _name>. To obtain more detailed information about the changes for invalidated objects, use the Details property.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleNotificationEventArgs Class](#)
- [OracleNotificationEventArgs Members](#)
- ["Details"](#)

Source

This property returns the database event source for the notification.

Declaration

```
// C#  
public OracleNotificationSource Source{get;}
```

Property Value

The OracleNotificationSource value for the notification.

Remarks

The OracleNotificationSource value is an enumeration type. If several event sources are received from the notification message, the Source property is set to one of the OracleNotificationSource enumeration values related to the database event source. For example, if a table has been altered (by the ALTER TABLE command) and a new row has been inserted into the same table, the Source property is set to either OracleNotificationSource.Object or OracleNotificationSource.Data.

For Continuous Query Notification:

- When the Source property is set to OracleNotificationSource.Data:
 - The Info property is set to one of the following:
 - * OracleNotificationInfo.Insert
 - * OracleNotificationInfo.Delete
 - * OracleNotificationInfo.Update
 - The ResourceNames property is set, and the elements are set to the invalidated object names.
 - The Details property contains detailed information on the change of each invalidated table.
- When the Source property is set to OracleNotificationSource.Database:
 - The Info property is set to one of the following:
 - * OracleNotificationInfo.Startup
 - * OracleNotificationInfo.Shutdown
 - * OracleNotificationInfo.Shutdown_Any
 - * OracleNotificationInfo.Dropped
- When the Source property is set to OracleNotificationSource.Object:
 - The Info property is set to either OracleNotificationInfo.Altered or OracleNotificationInfo.Dropped.
 - The ResourceNames property is set, and the array elements of the ResourceNames property are set to the object names that have been altered or dropped.
 - The Details property contains detailed information on the changes of the object.
- When the Source property is set to OracleNotificationSource.Subscription:
 - The Info property is set to the following:
 - * OracleNotificationInfo.End

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleNotificationEventArgs Class](#)
- [OracleNotificationEventArgs Members](#)
- ["OracleNotificationSource Enumeration"](#)

Type

This property returns the database event type for the notification.

Declaration

```
// C#
public OracleNotificationType Type{get;}
```

Property Value

An OracleNotificationType enumeration value that represents the type of the database event notification.

Remarks

The OracleNotificationType value is an enumeration type. If several event types are received from the notification message, then the Type property is set to one of the OracleNotificationType enumeration values related to the database event type.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleNotificationEventArgs Class](#)
- [OracleNotificationEventArgs Members](#)
- ["OracleNotificationType Enumeration"](#)

OracleNotificationEventArgs Methods

OracleNotificationEventArgs methods are listed in [Table 9-26](#).

Table 9-26 OracleNotificationEventArgs Methods

Methods	Description
Equals	Inherited from System.Object
GetHashCode	Inherited from System.Object
GetType	Inherited from System.Object
ToString	Inherited from System.Object

📘 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleNotificationEventArgs Class](#)
- [OracleNotificationEventArgs Members](#)

OnChangeEventHandler Delegate

The OnChangeEventHandler delegate represents the signature of the method that handles the notification.

Declaration

```
// C#
public delegate void OnChangeEventHandler(object sender,
    OracleNotificationEventArgs args);
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Parameters

- *sender*
The source of the event.
- *args*
The OracleNotificationEventArgs instance that contains the event data.

📘 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleNotificationEventArgs Class](#)
- [OracleNotificationEventArgs Members](#)

OracleRowidInfo Enumeration

OracleRowidInfo enumeration values specify whether ROWID information is included as part of the ChangeNotificationEventArgs or not.

[Table 9-28](#) lists all the OracleRowidInfo enumeration values with a description of each enumerated value.

Table 9-27 OracleRowidInfo Members

Member Name	Description
Default	ROWID information is included only if <code>OracleCommand.AddRowid</code> property is set to true or if ROWID column is explicitly included in the query.
Include	ROWID information is included regardless of whether ROWID is included in the select-list of the query or not.
Exclude	ROWID information is not included regardless of whether ROWID is included in the select-list of the query or not.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- "[RowidInfo](#)"

OracleNotificationType Enumeration

OracleNotificationType enumerated values specify the different types that cause the notification.

[Table 9-28](#) lists all the OracleNotificationType enumeration values with a description of each enumerated value.

Table 9-28 OracleNotificationType Members

Member Name	Description
Change	A change occurs in the database.
Subscribe	A change occurs in the subscription.
Query	A query-based change occurs in the database.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

See Also

"[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"

OracleNotificationSource Enumeration

OracleNotificationSource enumerated values specify the different sources that cause notification.

[Table 9-29](#) lists all the OracleNotificationSource enumeration values with a description of each enumerated value.

Table 9-29 OracleNotificationSource Members

Member Name	Description
Data	The data in a table has changed.
Database	A database event such as a database startup or shutdown occurs.
Object	A database object is altered or dropped.
Subscription	The subscription is changed.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

See Also

["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)

OracleNotificationInfo Enumeration

OracleNotificationInfo enumerated values specify the database event that causes the notification.

[Table 9-30](#) lists all the OracleNotificationInfo enumeration values with a description of each enumerated value.

Table 9-30 OracleNotificationInfo Members

Member Name	Description
Insert	A row is inserted.
Delete	A row is deleted.
Update	A row is updated.
Startup	A database starts.
Shutdown	A database shuts down.
Shutdown_any	A database instance in a Real Application Cluster (Oracle RAC) environment shuts down.
Alter	An object is altered.
Drop	An object or database is dropped.
End	A registration is removed.
Error	A notification error occurs.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

 See Also

["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)

Oracle Data Provider for .NET Globalization Classes

This chapter describes the ODP.NET globalization classes.

This chapter contains these topics:

- [OracleGlobalization Class](#)
- [OracleDatabaseCharset Enumeration](#)
- [OracleDatabaseNCharset Enumeration](#)

OracleGlobalization Class

The OracleGlobalization class is used to obtain and set the Oracle globalization settings of the session, thread, and local computer (read-only).

Class Inheritance

System.Object

Oracle.DataAccess.Client.OracleGlobalization

Declaration

public sealed class OracleGlobalization : ICloneable, IDisposable

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

An exception is thrown for invalid property values. All newly set property values are validated, except the TimeZone property.

Changing the OracleGlobalization object properties does not change the globalization settings of the session or the thread. Either the SetSessionInfo method of the OracleConnection object or the SetThreadInfo method of the OracleGlobalization object must be called to alter the session's and thread's globalization settings, respectively.

Example

```
// C#

using System;
using Oracle.DataAccess.Client;

class OracleGlobalizationSample
{
    static void Main()
    {
        // Get thread's globalization info
        OracleGlobalization glob = OracleGlobalization.GetThreadInfo();

        // Prints "glob.Language = AMERICAN"
        Console.WriteLine("glob.Language = " + glob.Language);

        // Set language on thread's globalization info
        glob.Language = "FRENCH";
        OracleGlobalization.SetThreadInfo(glob);
        OracleGlobalization.GetThreadInfo(glob);

        // Prints "glob.Language = FRENCH"
        Console.WriteLine("glob.Language = " + glob.Language);

        glob.Dispose();
    }
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleGlobalization Members](#)
- [OracleGlobalization Static Methods](#)
- [OracleGlobalization Properties](#)
- [OracleGlobalization Public Methods](#)
- *Oracle AI Database SQL Language Reference*
- *Oracle AI Database Globalization Support Guide*

OracleGlobalization Members

OracleGlobalization members are listed in the following tables.

OracleGlobalization Static Methods

The OracleGlobalization static methods are listed in [Table 10-1](#).

Table 10-1 OracleGlobalization Static Methods

Name	Description
GetClientInfo	Returns an OracleGlobalization object that represents the Oracle globalization settings of the local computer (Overloaded) <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
GetThreadInfo	Returns or refreshes an OracleGlobalization instance that represents Oracle globalization settings of the current thread (Overloaded) <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
SetThreadInfo	Sets Oracle globalization parameters to the current thread <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>

OracleGlobalization Properties

The OracleGlobalization properties are listed in [Table 10-2](#).

Table 10-2 OracleGlobalization Properties

Name	Description
Calendar	Specifies the calendar system
ClientCharacterSet	Specifies a client character set <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
Comparison	Specifies a method of comparison for WHERE clauses and comparison in PL/SQL blocks
Currency	Specifies the string to use as a local currency symbol for the L number format element
DateFormat	Specifies the date format for Oracle Date type as a string
DateLanguage	Specifies the language used to spell day and month names and date abbreviations
DualCurrency	Specifies the dual currency symbol, such as <i>Euro</i> , for the U number format element
ISOCurrency	Specifies the string to use as an international currency symbol for the C number format element
Language	Specifies the default language of the database
LengthSemantics	Enables creation of CHAR and VARCHAR2 columns using either byte or character (default) length semantics
NCharConversionException	Determines whether or not data loss during an implicit or explicit character type conversion reports an error
NumericCharacters	Specifies the characters used for the decimal character and the group separator character for numeric values in strings
Sort	Specifies the collating sequence for ORDER by clause
Territory	Specifies the name of the territory
TimeStampFormat	Specifies the string format for TimeStamp types
TimeStampTZFormat	Specifies the string format for TimeStampTZ types
TimeZone	Specifies the time zone region name

OracleGlobalization Public Methods

OracleGlobalization public methods are listed in [Table 10-3](#).

Table 10-3 OracleGlobalization Public Methods

Public Method	Description
Clone	Creates a copy of an OracleGlobalization object
Dispose	Releases any resources or memory allocated by the object

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)

OracleGlobalization Static Methods

The OracleGlobalization static methods are listed in [Table 10-4](#).

Table 10-4 OracleGlobalization Static Methods

Name	Description
GetClientInfo	Returns an OracleGlobalization object that represents the Oracle globalization settings of the local computer (Overloaded) <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
GetThreadInfo	Returns or refreshes an OracleGlobalization instance that represents Oracle globalization settings of the current thread (Overloaded) <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
SetThreadInfo	Sets Oracle globalization parameters to the current thread <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)

GetClientInfo

GetClientInfo returns an OracleGlobalization object instance that represents the Oracle globalization settings of the local computer.

Overload List:

- [GetClientInfo\(\)](#)
This method returns an OracleGlobalization instance that represents the globalization settings of the local computer.
- [GetClientInfo\(OracleGlobalization\)](#)
This method refreshes the provided OracleGlobalization object with the globalization settings of the local computer.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)

GetClientInfo()

This method returns an OracleGlobalization instance that represents the globalization settings of the local computer.

Declaration

```
// C#  
public static OracleGlobalization GetClientInfo();
```

Return Value

An OracleGlobalization instance.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Client;  
  
class GetClientInfoSample  
{  
    static void Main()  
    {  
        // Get client's globalization info  
        OracleGlobalization glob = OracleGlobalization.GetClientInfo();  
  
        // Prints "glob.Language = AMERICAN"  
        Console.WriteLine("glob.Language = " + glob.Language);  
  
        glob.Dispose();  
    }  
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)

GetClientInfo(OracleGlobalization)

This method refreshes the provided `OracleGlobalization` object with the globalization settings of the local computer.

Declaration

```
// C#  
public static void GetClientInfo(OracleGlobalization oraGlob);
```

Parameters

- *oraGlob*

The `OracleGlobalization` object being updated.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Client;  
  
class GetClientInfoSample  
{  
    static void Main()  
    {  
        // Get client's globalization info  
        OracleGlobalization glob = OracleGlobalization.GetClientInfo();  
  
        // Prints "glob.Language = AMERICAN"  
        Console.WriteLine("glob.Language = " + glob.Language);  
  
        // Get client's globalization info using overload  
        OracleGlobalization.GetClientInfo(glob);  
  
        // Prints "glob.Language = AMERICAN"  
        Console.WriteLine("glob.Language = " + glob.Language);  
  
        glob.Dispose();  
    }  
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)

GetThreadInfo

GetThreadInfo returns or refreshes an OracleGlobalization instance.

Overload List:

- [GetThreadInfo\(\)](#)
This method returns an OracleGlobalization object instance of the current thread.
- [GetThreadInfo\(OracleGlobalization\)](#)
This method refreshes the OracleGlobalization object instance with the globalization settings of the current thread.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)

GetThreadInfo()

This method returns an OracleGlobalization instance of the current thread.

Declaration

```
// C#  
public static OracleGlobalization GetThreadInfo();
```

Return Value

An OracleGlobalization instance.

Remarks

Initially, GetThreadInfo() returns an OracleGlobalization object that has the same property values as that returned by GetClientInfo(), unless the application changes it by invoking SetThreadInfo().

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Client;
```

```
class GetThreadInfoSample
{
    static void Main()
    {
        // Get thread's globalization info
        OracleGlobalization glob = OracleGlobalization.GetThreadInfo();

        // Prints "glob.Language = AMERICAN"
        Console.WriteLine("glob.Language = " + glob.Language);

        // Get thread's globalization info using overloaded
        OracleGlobalization.GetThreadInfo(glob);

        // Prints "glob.Language = AMERICAN"
        Console.WriteLine("glob.Language = " + glob.Language);

        glob.Dispose();
    }
}
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)

GetThreadInfo(OracleGlobalization)

This method refreshes the `OracleGlobalization` object with the globalization settings of the current thread.

Declaration

```
// C#
public static void GetThreadInfo(OracleGlobalization oraGlob);
```

Parameters

- *oraGlob*
The `OracleGlobalization` object being updated.

Remarks

Initially `GetThreadInfo()` returns an `OracleGlobalization` object that has the same property values as that returned by `GetClientInfo()`, unless the application changes it by invoking `SetThreadInfo()`.

Example

```
// C#

using System;
using Oracle.DataAccess.Client;

class GetThreadInfoSample
```

```
{
    static void Main()
    {
        // Get thread's globalization info
        OracleGlobalization glob = OracleGlobalization.GetThreadInfo();

        // Prints "glob.Language = AMERICAN"
        Console.WriteLine("glob.Language = " + glob.Language);

        // Get thread's globalization info using overloaded
        OracleGlobalization.GetThreadInfo(glob);

        // Prints "glob.Language = AMERICAN"
        Console.WriteLine("glob.Language = " + glob.Language);

        glob.Dispose();
    }
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)

SetThreadInfo

This method sets Oracle globalization parameters to the current thread.

Declaration

```
// C#
public static void SetThreadInfo(OracleGlobalization oraGlob);
```

Parameters

- *oraGlob*
An OracleGlobalization object.

Remarks

Any .NET string conversions to and from ODP.NET Types, as well as ODP.NET Type constructors, use the globalization property values where applicable. For example, when constructing an OracleDate structure from a .NET string, that string is expected to be in the format specified by the OracleGlobalization.DateFormat property of the thread.

Example

```
// C#

using System;
using Oracle.DataAccess.Client;

class SetThreadInfoSample
{
    static void Main()
```

```

{
    // Get thread's globalization info
    OracleGlobalization glob1 = OracleGlobalization.GetThreadInfo();

    // Prints "glob1.Language = AMERICAN"
    Console.WriteLine("glob1.Language = " + glob1.Language);

    // Set language on thread's globalization info
    glob1.Language = "FRENCH";
    OracleGlobalization.SetThreadInfo(glob1);
    OracleGlobalization glob2 = OracleGlobalization.GetThreadInfo();

    // Prints "glob2.Language = FRENCH"
    Console.WriteLine("glob2.Language = " + glob2.Language);

    glob1.Dispose();
    glob2.Dispose();
}
}

```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)

OracleGlobalization Properties

The OracleGlobalization properties are listed in [Table 10-5](#).

Table 10-5 OracleGlobalization Properties

Name	Description
Calendar	Specifies the calendar system
ClientCharacterSet	Specifies a client character set <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
Comparison	Specifies a method of comparison for WHERE clauses and comparison in PL/SQL blocks
Currency	Specifies the string to use as a local currency symbol for the L number format element
DateFormat	Specifies the date format for Oracle Date type as a string
DateLanguage	Specifies the language used to spell day and month names and date abbreviations
DualCurrency	Specifies the dual currency symbol, such as <i>Euro</i> , for the U number format element
ISOCurrency	Specifies the string to use as an international currency symbol for the C number format element
Language	Specifies the default language of the database

Table 10-5 (Cont.) OracleGlobalization Properties

Name	Description
LengthSemantics	Enables creation of CHAR and VARCHAR2 columns using either byte or character (default) length semantics
NCharConversionException	Determines whether or not data loss during an implicit or explicit character type conversion reports an error
NumericCharacters	Specifies the characters used for the decimal character and the group separator character for numeric values in strings
Sort	Specifies the collating sequence for ORDER by clause
Territory	Specifies the name of the territory
TimeStampFormat	Specifies the string format for TimeStamp types
TimeStampTZFormat	Specifies the string format for TimeStampTZ types
TimeZone	Specifies the time zone region name

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)

Calendar

This property specifies the calendar system.

Declaration

```
// C#  
public string Calendar {get; set;}
```

Property Value

A string representing the Calendar.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

The default value is the NLS_CALENDAR setting of the local computer. This value is the same regardless of whether or not the OracleGlobalization object represents the settings of the client, thread, or session.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)

ClientCharacterSet

This property specifies a client character set.

Declaration

```
// C#  
public string ClientCharacterSet {get;}
```

Property Value

A string that the provides the name of the character set of the local computer.

Remarks

The default value is the character set of the local computer.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)

Comparison

This property represents a method of comparison for WHERE clauses and comparison in PL/SQL blocks.

Declaration

```
// C#  
public string Comparison {get; set;}
```

Property Value

A string that provides the name of the method of comparison.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

The default value is the NLS_COMP setting of the local computer.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)

Currency

This property specifies the string to use as a local currency symbol for the L number format element.

Declaration

```
// C#  
public string Currency {get; set;}
```

Property Value

The string to use as a local currency symbol for the L number format element.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

The default value is the NLS_CURRENCY setting of the local computer.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)
- *Oracle AI Database SQL Language Reference* for further information on the L number format element

DateFormat

This property specifies the date format for Oracle Date type as a string.

Declaration

```
// C#  
public string DateFormat {get; set;}
```

Property Value

The date format for Oracle Date type as a string

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

The default value is the NLS_DATE_FORMAT setting of the local computer.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)

DateLanguage

This property specifies the language used to spell names of days and months, and date abbreviations (for example: a.m., p.m., AD, BC).

Declaration

```
// C#  
public string DateLanguage {get; set;}
```

Property Value

A string specifying the language.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

The default value is the NLS_DATE_LANGUAGE setting of the local computer.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)

DualCurrency

This property specifies the dual currency symbol, such as *Euro*, for the U number format element.

Declaration

```
// C#  
public string DualCurrency {get; set;}
```

Property Value

A string that provides the dual currency symbol.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

The default value is the NLS_DUAL_CURRENCY setting of the local computer.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)
- *Oracle AI Database SQL Language Reference* for further information on the U number format element

ISOCurrency

This property specifies the string to use as an international currency symbol for the C number format element.

Declaration

```
// C#  
public string ISOCurrency {get; set;}
```

Property Value

The string used as an international currency symbol.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

The default value is the NLS_ISO_CURRENCY setting of the local computer.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)
- *Oracle AI Database SQL Language Reference* for further information on the C number format element

Language

This property specifies the default language of the database.

Declaration

```
// C#  
public string Language {get; set;}
```

Property Value

The default language of the database.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

The default value is the NLS_LANGUAGE setting of the local computer.

Language is used for messages, day and month names, and sorting algorithms. It also determines NLS_DATE_LANGUAGE and NLS_SORT parameter values.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)

LengthSemantics

This property indicates whether or not CHAR and VARCHAR2 columns use byte or character (default) length semantics.

Declaration

```
// C#  
public string LengthSemantics {get; set;}
```

Property Value

A string that indicates either byte or character length semantics.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

The default value is the NLS_LENGTH_SEMANTICS setting of the local computer.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)

NCharConversionException

This property determines whether or not data loss during an implicit or explicit character type conversion reports an error.

Declaration

```
// C#  
public bool NCharConversionException {get; set;}
```

Property Value

A string that indicates whether or not a character type conversion causes an error message.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

The default value of NLS_NCHAR_CONV_EXCP is False, unless it is overridden by a setting in the INIT.ORA file.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)

NumericCharacters

This property specifies the characters used for the decimal character and the group separator character for numeric values in strings.

Declaration

```
// C#  
public string NumericCharacters {get; set;}
```

Property Value

A string that represents the characters used.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

The default value is the NLS_NUMERIC_CHARACTERS setting of the local computer.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)

Sort

This property specifies the collating sequence for ORDER by clause.

Declaration

```
// C#  
public string Sort {get; set;}
```

Property Value

A string that indicates the collating sequence.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

The default value is the NLS_SORT setting of the local computer.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)

Territory

This property specifies the name of the territory.

Declaration

```
// C#  
public string Territory {get; set;}
```

Property Value

A string that provides the name of the territory.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

The default value is the NLS_TERRITORY setting of the local computer.

Changing this property changes other globalization properties.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)
- *Oracle AI Database Globalization Support Guide.*

TimeStampFormat

This property specifies the string format for TimeStamp types.

Declaration

```
// C#  
public string TimeStampFormat {get; set;}
```

Property Value

The string format for TimeStamp types.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

The default value is the NLS_TIMESTAMP_FORMAT setting of the local computer.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)

TimeStampTZFormat

This property specifies the string format for TimeStampTZ types.

Declaration

```
// C#  
public string TimeStampTZFormat {get; set;}
```

Property Value

The string format for TimeStampTZ types.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

The default value is the NLS_TIMESTAMP_TZ_FORMAT setting of the local computer.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)

TimeZone

This property specifies the time zone region name or hour offset.

Declaration

```
// C#  
public string TimeZone {get; set;}
```


Property Value

The string represents the time zone region name or the time zone offset.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the time zone region name of the local computer

`TimeZone` is only used when the thread constructs one of the `TimeStamp` structures. `TimeZone` has no effect on the session.

`TimeZone` can be either an hour offset, for example, 7:00, or a valid time zone region name that is provided in `V$TIMEZONE_NAMES`, such as `US/Pacific`. Time zone abbreviations are not supported.

Note

PST is a time zone region name as well as a time zone abbreviation; therefore it is accepted by `OracleGlobalization`.

This property returns an empty string if the `OracleGlobalization` object is obtained using `GetSessionInfo()` or `GetSessionInfo(OracleGlobalization)`. Initially, by default, the time zone of the session is identical to the time zone of the thread. Therefore, given that the session time zone is not changed by invoking `ALTER SESSION` calls, the session time zone can be fetched from the client's globalization settings.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)

OracleGlobalization Public Methods

`OracleGlobalization` public methods are listed in [Table 10-6](#).

Table 10-6 OracleGlobalization Public Methods

Public Method	Description
Clone	Creates a copy of an <code>OracleGlobalization</code> object
Dispose	Releases any resources or memory allocated by the object

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)

Clone

This method creates a copy of an OracleGlobalization object.

Declaration

```
// C#  
public object Clone();
```

Return Value

An OracleGlobalization object.

Implements

ICloneable

Remarks

The cloned object has the same property values as that of the object being cloned.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)

Dispose

This method releases any resources or memory allocated by the object.

Declaration

```
// C#  
public void Dispose();
```

Implements

IDisposable

Remarks

The Dispose method also closes the OracleGlobalization object.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleGlobalization Class](#)
- [OracleGlobalization Members](#)

OracleDatabaseCharset Enumeration

The OracleDatabaseCharset enumeration specifies the Oracle CHAR character set.

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

Table 10-7 OracleDatabaseCharset Enumeration Member Name and Description

Member Name	Description
AL32UTF8	Unicode 9.0 Universal Character Set (UCS), UTF-8 encoding scheme
AR8AD0S710	Arabic MS-DOS 710 Server 8-bit Latin/Arabic
AR8AD0S720	Arabic MS-DOS 720 Server 8-bit Latin/Arabic
AR8APTEC715	APTEC 715 Server 8-bit Latin/Arabic
AR8ARABICMACS	Mac Server 8-bit Latin/Arabic
AR8ASM08X	ASMO Extended 708 8-bit Latin/Arabic
AR8EBDIC420S	EBCDIC Code Page 420 Server 8-bit Latin/Arabic
AR8EBDICX	EBCDIC XBASIC Server 8-bit Latin/Arabic
AR8IS08859P6	ISO 8859-6 Latin/Arabic
AR8MSWIN1256	MS Windows Code Page 1256 8-Bit Latin/Arabic
AR8MUSSAD768	Mussa'd Alarabi/2 768 Server 8-bit Latin/Arabic
AR8NAFITHA711	Nafitha Enhanced 711 Server 8-bit Latin/Arabic
AR8NAFITHA721	Nafitha International 721 Server 8-bit Latin/Arabic
AR8SAKHR706	SAKHR 706 Server 8-bit Latin/Arabic
AR8SAKHR707	SAKHR 707 Server 8-bit Latin/Arabic
AZ8IS08859P9E	ISO 8859-9 Latin Azerbaijani
BG8MSWIN	MS Windows 8-bit Bulgarian Cyrillic

Table 10-7 (Cont.) OracleDatabaseCharset Enumeration Member Name and Description

Member Name	Description
BG8PC437S	IBM-PC Code Page 437 8-bit (Bulgarian Modification)
BLT8CP921	Latvian Standard LVS8-92(1) Windows/Unix 8-bit Baltic
BLT8EBCDIC1112	EBCDIC Code Page 1112 8-bit Server Baltic Multilingual
BLT8EBCDIC1112S	EBCDIC Code Page 1112 8-bit Server Baltic Multilingual
BLT8ISO8859P13	ISO 8859-13 Baltic
BLT8MSWIN1257	MS Windows Code Page 1257 8-bit Baltic
BLT8PC775	IBM-PC Code Page 775 8-bit Baltic
BN8BSCII	Bangladesh National Code 8-bit BSCII
CDN8PC863	IBM-PC Code Page 863 8-bit Canadian French
CE8BS2000	Siemens EBCDIC.DF.04 8-bit Central European
CEL8ISO8859P14	ISO 8859-14 Celtic
CL8BS2000	Siemens EBCDIC.EHC.LC 8-bit Cyrillic
CL8EBCDIC1025	EBCDIC Code Page 1025 Server 8-bit Cyrillic
CL8EBCDIC1025C	EBCDIC Code Page 1025 Client 8-bit Cyrillic
CL8EBCDIC1025R	EBCDIC Code Page 1025 Server 8-bit Cyrillic
CL8EBCDIC1025S	EBCDIC Code Page 1025 Server 8-bit Cyrillic
CL8EBCDIC1025X	EBCDIC Code Page 1025 (Modified) 8-bit Cyrillic
CL8EBCDIC1158	EBCDIC Code Page 1158 8-bit Cyrillic
CL8EBCDIC1158R	EBCDIC Code Page 1158 Server 8-bit Cyrillic
CL8ISO8859P5	ISO 8859-5 Latin/Cyrillic
CL8ISOIR111	ISOIR111 Cyrillic
CL8KOI8R	RELCOM Internet Standard 8-bit Latin/Cyrillic
CL8KOI8U	KOI8 Ukrainian Cyrillic
CL8MACCYRILLICS	Mac Server 8-bit Latin/Cyrillic
CL8MSWIN1251	MS Windows Code Page 1251 8-bit Latin/Cyrillic
D8BS2000	Siemens 9750-62 EBCDIC 8-bit German
D8EBCDIC1141	EBCDIC Code Page 1141 8-bit Austrian German
D8EBCDIC273	EBCDIC Code Page 273/1 8-bit Austrian German
DK8BS2000	Siemens 9750-62 EBCDIC 8-bit Danish
DK8EBCDIC1142	EBCDIC Code Page 1142 8-bit Danish
DK8EBCDIC277	EBCDIC Code Page 277/1 8-bit Danish
E8BS2000	Siemens 9750-62 EBCDIC 8-bit Spanish
EE8BS2000	Siemens EBCDIC.DF.04 8-bit East European
EE8EBCDIC870	EBCDIC Code Page 870 8-bit East European

Table 10-7 (Cont.) OracleDatabaseCharset Enumeration Member Name and Description

Member Name	Description
EE8EBCDIC870C	EBCDIC Code Page 870 Client 8-bit East European
EE8EBCDIC870S	EBCDIC Code Page 870 Server 8-bit East European
EE8IS08859P2	ISO 8859-2 East European
EE8MACCES	Mac Server 8-bit Central European
EE8MACCROATIANS	Mac Server 8-bit Croatian
EE8MSWIN1250	MS Windows Code Page 1250 8-bit East European
EE8PC852	IBM-PC Code Page 852 8-bit East European
EEC8EUROASCI	EEC Targon 35 ASCII West European/Greek
EEC8EUROPA3	EEC EUROPA3 8-bit West European/Greek
EL8DEC	DEC 8-bit Latin/Greek
EL8EBCDIC423R	IBM EBCDIC Code Page 423 for RDBMS server-side
EL8EBCDIC875	EBCDIC Code Page 875 8-bit Greek
EL8EBCDIC875R	EBCDIC Code Page 875 Server 8-bit Greek
EL8EBCDIC875S	Obsolete. Repalced by EL8EBCDIC875S
EL8GCOS7	Bull EBCDIC GCOS7 8-bit Greek
EL8IS08859P7	ISO 8859-7 Latin/Greek
EL8MACGREEKS	Mac Server 8-bit Greek
EL8MSWIN1253	MS Windows Code Page 1253 8-bit Latin/Greek
EL8PC437S	IBM-PC Code Page 437 8-bit (Greek modification)
EL8PC851	IBM-PC Code Page 851 8-bit Greek/Latin
EL8PC869	IBM-PC Code Page 869 8-bit Greek/Latin
ET8MSWIN923	MS Windows Code Page 923 8-bit Estonian
F8BS2000	Siemens 9750-62 EBCDIC 8-bit French
F8EBCDIC1147	EBCDIC Code Page 1147 8-bit French
F8EBCDIC297	EBCDIC Code Page 297 8-bit French
HU8ABMOD	Hungarian 8-bit Special AB Mod
HU8CWI2	Hungarian 8-bit CWI-2
I8EBCDIC1144	EBCDIC Code Page 1144 8-bit Italian
I8EBCDIC280	EBCDIC Code Page 280/1 8-bit Italian
IN8ISCII	Multiple-Script Indian Standard 8-bit Latin/Indian Languages
IS8PC861	IBM-PC Code Page 861 8-bit Icelandic
IW8EBCDIC1086	EBCDIC Code Page 1086 8-bit Hebrew
IW8EBCDIC424	EBCDIC Code Page 424 8-bit Latin/Hebrew
IW8EBCDIC424S	EBCDIC Code Page 424 Server 8-bit Latin/Hebrew

Table 10-7 (Cont.) OracleDatabaseCharset Enumeration Member Name and Description

Member Name	Description
IW8ISO8859P8	ISO 8859-8 Latin/Hebrew
IW8MACHEBREWS	Mac Server 8-bit Hebrew
IW8MSWIN1255	MS Windows Code Page 1255 8-bit Latin/Hebrew
IW8PC1507	IBM-PC Code Page 1507/862 8-bit Latin/Hebrew
JA16DBCS	IBM EBCDIC 16-bit Japanese
JA16EBDIC930	IBM DBCS Code Page 290 16-bit Japanese
JA16EUC	EUC 24-bit Japanese
JA16EUCTILDE	The same as JA16EUC except for the way that the wave dash and the tilde are mapped to and from Unicode.
JA16SJIS	Shift-JIS 16-bit Japanese
JA16SJISTILDE	The same as JA16SJIS except for the way that the wave dash and the tilde are mapped to and from Unicode.
JA16VMS	JVMS 16-bit Japanese
K016DBCS	IBM EBCDIC 16-bit Korean
K016KSC5601	KSC5601 16-bit Korean
K016KSCCS	KSCCS 16-bit (Johab) Korean
K016MSWIN949	MS Windows Code Page 949 Korean
LA8ISO6937	ISO 6937 8-bit Coded Character Set for Text Communication
LA8PASSPORT	German Government Printer 8-bit All-European Latin
LT8MSWIN921	MS Windows Code Page 921 8-bit Lithuanian
LT8PC772	IBM-PC Code Page 772 8-bit Lithuanian (Latin/Cyrillic)
LT8PC774	IBM-PC Code Page 774 8-bit Lithuanian (Latin)
LV8PC1117	IBM-PC Code Page 1117 8-bit Latvian
LV8PC8LR	Latvian Version IBM-PC Code Page 866 8-bit Latin/Cyrillic
LV8RST104090	IBM-PC Alternative Code Page 8-bit Latvian (Latin/Cyrillic)
N8PC865	IBM-PC Code Page 865 8-bit Norwegian
NE8ISO8859P10	ISO 8859-10 North European
NEE8ISO8859P4	ISO 8859-4 North and North-East European
RU8BESTA	BESTA 8-bit Latin/Cyrillic
RU8PC855	IBM-PC Code Page 855 8-bit Latin/Cyrillic
RU8PC866	IBM-PC Code Page 866 8-bit Latin/Cyrillic
S8BS2000	Siemens 9750-62 EBCDIC 8-bit Swedish
S8EBDIC1143	EBCDIC Code Page 1143 8-bit Swedish

Table 10-7 (Cont.) OracleDatabaseCharset Enumeration Member Name and Description

Member Name	Description
S8EBCDIC278	EBCDIC Code Page 278/1 8-bit Swedish
SE8ISO8859P3	ISO 8859-3 South European
TH8MACTHAI	Mac Client 8-bit Latin/Thai
TH8TISASCII	Thai Industrial Standard 620-2533 - ASCII 8-bit
TH8TISEBCDIC	Thai Industrial Standard 620-2533 - EBCDIC 8-bit
TH8TISEBCDICS	Thai Industrial Standard 620-2533-EBCDIC Server 8-bit
TIMESTEN8	TimesTen Legacy
TR8DEC	DEC 8-bit Turkish
TR8EBCDIC1026	EBCDIC Code Page 1026 8-bit Turkish
TR8EBCDIC1026S	EBCDIC Code Page 1026 Server 8-bit Turkish
TR8MACTURKISHS	Mac Server 8-bit Turkish
TR8MSWIN1254	MS Windows Code Page 1254 8-bit Turkish
TR8PC857	IBM-PC Code Page 857 8-bit Turkish
US7ASCII	ASCII 7-bit American
US8BS2000	Siemens 9750-62 EBCDIC 8-bit American
US8ICL	ICL EBCDIC 8-bit American
US8PC437	IBM-PC Code Page 437 8-bit American
VN8MSWIN1258	MS Windows Code Page 1258 8-bit Vietnamese
VN8VN3	VN3 8-bit Vietnamese
WE8BS2000	Siemens EBCDIC.DF.04 8-bit West European
WE8BS2000E	Siemens EBCDIC.DF.04 8-bit West European
WE8BS2000L5	Siemens EBCDIC.DF.L5 8-bit West European/ Turkish
WE8DEC	DEC 8-bit West European
WE8DG	DG 8-bit West European
WE8EBCDIC1047	EBCDIC Code Page 1047 8-bit West European
WE8EBCDIC1047E	Latin 1/Open Systems 1047
WE8EBCDIC1140	EBCDIC Code Page 1140 8-bit West European
WE8EBCDIC1140C	EBCDIC Code Page 1140 8-bit West European
WE8EBCDIC1145	EBCDIC Code Page 1145 8-bit West European
WE8EBCDIC1146	EBCDIC Code Page 1146 8-bit West European
WE8EBCDIC1148	EBCDIC Code Page 1148 8-bit West European
WE8EBCDIC1148C	EBCDIC Code Page 1148 Client 8-bit West European
WE8EBCDIC284	EBCDIC Code Page 284 8-bit Latin American/ Spanish
WE8EBCDIC285	EBCDIC Code Page 285 8-bit West European

Table 10-7 (Cont.) OracleDatabaseCharset Enumeration Member Name and Description

Member Name	Description
WE8EBCDIC37	EBCDIC Code Page 37 8-bit West European
WE8EBCDIC37C	EBCDIC Code Page 37 8-bit Oracle/c
WE8EBCDIC500	EBCDIC Code Page 500 8-bit West European
WE8EBCDIC500C	EBCDIC Code Page 500 8-bit Oracle/c
WE8EBCDIC871	EBCDIC Code Page 871 8-bit Icelandic
WE8EBCDIC924	Latin 9 EBCDIC 924
WE8GCOS7	Bull EBCDIC GCOS7 8-bit West European
WE8ICL	ICL EBCDIC 8-bit West European
WE8ISO8859P1	ISO 8859-1 West European
WE8ISO8859P15	ISO 8859-15 West European
WE8ISO8859P9	ISO 8859-9 West European & Turkish
WE8MACROMAN8	Mac Client 8-bit Extended Roman8 West European
WE8MSWIN1252	MS Windows Code Page 1252 8-bit West European
WE8NCR4970	NCR 4970 8-bit West European
WE8NEXTSTEP	NeXTSTEP PostScript 8-bit West European
WE8PC850	IBM-PC Code Page 850 8-bit West European
WE8PC858	IBM-PC Code Page 858 8-bit West European
WE8PC860	IBM-PC Code Page 860 8-bit West European
WE8ROMAN8	HP Roman8 8-bit West European
ZHS16CGB231280	CGB2312-80 16-bit Simplified Chinese
ZHS16DBCS	IBM EBCDIC 16-bit Simplified Chinese
ZHS16GBK	GBK 16-bit Simplified Chinese
ZHS32GB18030	GB18030 32-bit Simplified Chinese
ZHT16BIG5	BIG5 16-bit Traditional Chinese
ZHT16CCDC	HP CCDC 16-bit Traditional Chinese
ZHT16DBCS	IBM EBCDIC 16-bit Traditional Chinese
ZHT16DBT	Taiwan Taxation 16-bit Traditional Chinese
ZHT16HKSCS	MS Windows Code Page 950 with Hong Kong Supplementary Character Set HKSCS-2001 (character set conversion to and from Unicode is based on Unicode 3.0)
ZHT16HKSCS31	MS Windows Code Page 950 with Hong Kong Supplementary Character Set HKSCS-2001 (character set conversion to and from Unicode is based on Unicode 3.1)
ZHT16MSWIN950	MS Windows Code Page 950 Traditional Chinese
ZHT32EUC	EUC 32-bit Traditional Chinese
ZHT32SOPS	SOPS 32-bit Traditional Chinese

Table 10-7 (Cont.) OracleDatabaseCharset Enumeration Member Name and Description

Member Name	Description
ZHT32TRIS	TRIS 32-bit Traditional Chinese

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- Recommended Database Character Sets in the *Database Globalization Support Guide* for more information about the Member Names and Descriptions
- [OracleGlobalization Class](#)

OracleDatabaseNCharset Enumeration

The OracleDatabaseNCharset enumeration specifies the Oracle NCHAR character set. Possible values for this enumeration are:

Table 10-8 OracleDatabaseNCharset Enumeration

Member Name	Description
AL16UTF16	Unicode 15.0 Universal character set, UTF-16BE encoding scheme
UTF8	Unicode 3.0 Universal character set, CESU-8 encoding scheme

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleGlobalization Class](#)

Oracle Data Provider for .NET Transparent Application Failover Classes

This chapter describes the ODP.NET Transparent Application Failover(TAF) classes and enumerations.

This chapter contains these topics:

- [OracleFailoverEventArgs Class](#)
- [OracleTAFMode Class](#)
- [OracleFailoverEventHandler Delegate](#)
- [FailoverEvent Enumeration](#)
- [FailoverReturnCode Enumeration](#)
- [FailoverType Enumeration](#)
- [OracleFailoverEvent Enumeration](#)
- [OracleFailoverRestore Enumeration](#)
- [OracleFailoverReturnCode Enumeration](#)
- [OracleFailoverType Enumeration](#)

OracleFailoverEventArgs Class

The `OracleFailoverEventArgs` class provides event data for the `OracleConnection.Failover` event. When database failover occurs, the `OracleConnection.Failover` event is triggered along with the `OracleFailoverEventArgs` object that stores the event data.

Class Inheritance

`System.Object`

`System.EventArgs`

`Oracle.DataAccess.Client.OracleFailoverEventArgs`

Declaration

```
// C#
public sealed class OracleFailoverEventArgs
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example (Oracle.DataAccess.Client only)

```
// Transparent Application Failover (TAF) Setup
// Refer Oracle® Database Net Services Administrator's Guide

// C#

using System;
using System.Threading;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class FailoverSample
{
    static void Main(string[] args)
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        // Register the event handler OnFailover
        con.Failover += new OracleFailoverEventHandler(OnFailover);

        Console.WriteLine("Wait for a failover for 5 seconds");
        Thread.Sleep(5000);

        con.Close();
        con.Dispose();
    }

    // TAF callback function
    static FailoverReturnCode OnFailover(object sender,
        OracleFailoverEventArgs eventArgs)
    {
        switch (eventArgs.FailoverEvent)
        {
            case FailoverEvent.Begin:
            {
                Console.WriteLine("FailoverEvent.Begin - Failover is starting");
                Console.WriteLine("FailoverType = " + eventArgs.FailoverType);
                break;
            }
            case FailoverEvent.End:
            {
                Console.WriteLine("FailoverEvent.End - Failover was successful");
            }
        }
    }
}
```

```
        break;
    }
    case FailoverEvent.Reauth:
    {
        Console.WriteLine("FailoverEvent.Reauth - User reauthenticated");
        break;
    }
    case FailoverEvent.Error:
    {
        Console.WriteLine("FailoverEvent.Error - Failover was unsuccessful");

        // Sleep for 3 sec and Retry
        Thread.Sleep(3000);
        return FailoverReturnCode.Retry;
    }
    case FailoverEvent.Abort:
    {
        Console.WriteLine("FailoverEvent.Abort - Failover was unsuccessful");
        break;
    }
    default:
    {
        Console.WriteLine("Invalid FailoverEvent : " + eventArgs.FailoverEvent);
        break;
    }
}
return FailoverReturnCode.Success;
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleFailoverEventArgs Members](#)
- [OracleFailoverEventArgs Static Methods](#)
- [OracleFailoverEventArgs Properties](#)
- [OracleFailoverEventArgs Public Methods](#)
- ["OracleConnection Class"](#)
- *Oracle AI Database Net Services Administrator's Guide*

OracleFailoverEventArgs Members

OracleFailoverEventArgs members are listed in the following tables.

OracleFailoverEventArgs Static Methods

The OracleFailoverEventArgs static methods are listed in [Table 11-1](#).

Table 11-1 OracleFailoverEventArgs Static Methods

Methods	Description
Equals	Inherited from System.Object (Overloaded)

OracleFailoverEventArgs Properties

The OracleFailoverEventArgs properties are listed in [Table 11-2](#).

Table 11-2 OracleFailoverEventArgs Properties

Name	Description
FailoverEvent	Indicates the state of the failover
FailoverType	Specifies the type of failover the client has requested
OracleFailoverEvent	Indicates the state of the failover <i>Not available in ODP.NET, Unmanaged Driver</i>
OracleFailoverType	Specifies the type of failover the client has requested <i>Not available in ODP.NET, Unmanaged Driver</i>

OracleFailoverEventArgs Public Methods

The OracleFailoverEventArgs public methods are listed in [Table 11-3](#).

Table 11-3 OracleFailoverEventArgs Public Methods

Name	Description
Equals	Inherited from System.Object (Overloaded)
GetHashCode	Inherited from System.Object
GetType	Inherited from System.Object
ToString	Inherited from System.Object

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleFailoverEventArgs Class](#)
- [FailoverType Enumeration](#)
- [OracleFailoverType Enumeration](#)

OracleFailoverEventArgs Static Methods

The OracleFailoverEventArgs static methods are listed in [Table 11-1](#).

Table 11-4 OracleFailoverEventArgs Static Methods

Methods	Description
Equals	Inherited from System.Object (Overloaded)

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleFailoverEventArgs Class](#)
- [OracleFailoverEventArgs Members](#)

OracleFailoverEventArgs Properties

The OracleFailoverEventArgs properties are listed in [Table 11-5](#).

Table 11-5 OracleFailoverEventArgs Properties

Name	Description
FailoverEvent	Indicates the state of the failover
FailoverType	Specifies the type of failover the client has requested
OracleFailoverEvent	Indicates the state of the failover <i>Not available in ODP.NET, Unmanaged Driver</i>
OracleFailoverType	Specifies the type of failover the client has requested <i>Not available in ODP.NET, Unmanaged Driver</i>

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleFailoverEventArgs Class](#)
- [OracleFailoverEventArgs Members](#)

FailoverEvent

This property indicates the state of the failover.

Declaration

```
// C#
public FailoverEvent FailoverEvent {get;}
```

Property Value

A FailoverEvent enumerated value.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleFailoverEventArgs Class](#)
- [OracleFailoverEventArgs Members](#)
- ["FailoverEvent Enumeration"](#)

FailoverType

This property indicates the state of the failover.

Declaration

```
// C#  
public FailoverType FailoverType {get;}
```

Property Value

A FailoverType enumeration value.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleFailoverEventArgs Class](#)
- [OracleFailoverEventArgs Members](#)
- ["FailoverType Enumeration"](#)

OracleFailoverEvent

This property indicates the state of the failover.

Declaration

```
// C#  
public OracleFailoverEvent FailoverEvent {get;}
```

Property Value

A OracleFailoverEvent enumerated value.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleFailoverEventArgs Class](#)
- [OracleFailoverEventArgs Members](#)
- [OracleFailoverEvent Enumeration](#)

OracleFailoverType

This property indicates the state of the failover.

Declaration

```
// C#
public OracleFailoverType FailoverType {get;}
```

Property Value

A OracleFailoverType enumeration value.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleFailoverEventArgs Class](#)
- [OracleFailoverEventArgs Members](#)
- [OracleFailoverType Enumeration](#)

OracleFailoverEventArgs Public Methods

The OracleFailoverEventArgs public methods are listed in [Table 11-6](#).

Table 11-6 OracleFailoverEventArgs Public Methods

Name	Description
Equals	Inherited from System.Object (Overloaded)
GetHashCode	Inherited from System.Object
GetType	Inherited from System.Object
ToString	Inherited from System.Object

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleFailoverEventArgs Class](#)
- [OracleFailoverEventArgs Members](#)

OracleTAFMode Class

An OracleTAFMode object sets the Oracle Transparent Application Failover parameters for the connection.

Class Inheritance

System.Object
Oracle.ManagedDataAccess.OracleTAFMode

Declaration

```
// C#  
public sealed class OracleTAFMode
```

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

Thread Safety

All public static methods are thread-safe.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleTAFMode Constructors](#)

OracleTAFMode Constructors

OracleTAFMode constructors instantiates new instances of the OracleTAFMode class.

Overload List:

- `OracleTAFMode()`
This constructor instantiates a new instance of the `OracleTAFMode` class using default property values.
- `OracleTAFMode(OracleFailoverType, OracleFailoverRestore)`
This constructor instantiates a new instance of the `OracleTAFMode` class using the provided failover type and restore level.
- `OracleTAFMode(OracleFailoverType, OracleFailoverRestore, int, int)`
This constructor instantiates a new instance of the `OracleTAFMode` class using the provided failover type, restore level, retries, and delay.

Arguments:

- `failoverType`: Specifies the type of failover. Possible values are:
 - `NONE`: This is the default. No failover functionality is used. This can also be explicitly specified to prevent failover from happening.
 - `SESSION`: Fails over the session. For example, if a user's connection is lost, then a new session is automatically created for the user. This type of failover does not attempt to recover selects.
- `failoverRestore`: Specifies the session restore level. Possible values are:
 - `NONE`: This is the default. No session state will be restored.
 - `LEVEL1`: Restores basic session state along with NLS parameters.
- `failoverRetries`: Specifies the number of times to attempt to connect after a failover. Default value is 0.
- `failoverDelay`: Specifies the amount of time in seconds to wait between connect attempts. Default value is 0.

Exception

`ArgumentException` is thrown if negative value is provided for `failoverRetries` or `failoverDelay` arguments in the constructor.

① See Also

- [OracleTAFMode Class](#)

OracleFailoverEventHandler Delegate

The `OracleFailoverEventHandler` represents the signature of the method that handles the `OracleConnection.Failover` event.

Declaration

```
// C#
public delegate FailoverReturnCode OracleFailoverEventHandler(object sender,
    OracleFailoverEventArgs eventArgs);
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Parameter

- *sender*
The source of the event.
- *eventArgs*
The OracleFailoverEventArgs object that contains the event data.

Return Type

An int.

Remarks

To receive failover notifications, a callback function can be registered as follows:

```
ConObj.Failover += new OracleFailoverEventHandler(OnFailover);
```

The definition of the callback function OnFailover can be as follows:

```
public FailoverReturnCode OnFailover(object sender, OracleFailoverEventArgs eventArgs)
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleFailoverEventArgs Class](#)
- [OracleFailoverEventArgs Members](#)
- ["Failover"](#)

FailoverEvent Enumeration

FailoverEvent enumerated values are used to specify the state of the failover.

[Table 11-7](#) lists all the FailoverEvent enumeration values with a description of each enumerated value.

Table 11-7 FailoverEvent Enumeration Values

Member Names	Description
<code>FailoverEvent.Begin</code>	Indicates that failover has detected a lost connection and that failover is starting.
<code>FailoverEvent.End</code>	Indicates successful completion of failover.
<code>FailoverEvent.Abort</code>	Indicates that failover was unsuccessful, and there is no option of retrying.
<code>FailoverEvent.Error</code>	Indicates that failover was unsuccessful, and it gives the application the opportunity to handle the error and retry failover. The application can retry failover by returning <code>FailoverReturnCode.Retry</code> for the event notification.
<code>FailoverEvent.Reauth</code>	Indicates that a user handle has been reauthenticated. This applies to the situation where a client has multiple user sessions on a single server connection. During the initial failover, only the active user session is failed over. Other sessions are failed over when the application tries to use them. This is the value passed to the callback during these subsequent failovers.

No significant database operation should occur immediately after a `FailoverEvent.Begin` event. SQL and major database operations should wait until the `FailoverEvent.End` event. `FailoverEvent.Begin` is primarily used to reject failover or to trace it. `FailoverEvent.Begin` can also be used for non-database application operations, such as informing the end user a failover is in progress and to wait until it completes before proceeding. Transactions can be used in the `FailoverEvent.End` callback phase, such as to file fault tickets or audit. These transactions must be committed before the callback completes.

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	<code>Oracle.DataAccess.dll</code>
Namespace	<code>Oracle.DataAccess.Client</code>
.NET Framework	See System Requirements

See Also

- [FailoverEvent Enumeration](#)
- ["OracleFailoverEventArgs Class"](#)
- ["FailoverEvent"](#)
- *Oracle Real Application Clusters Administration and Deployment Guide*
- *Oracle AI Database Net Services Reference*

FailoverReturnCode Enumeration

FailoverReturnCode enumerated values are passed back by the application to the ODP.NET provider to request a retry in case of a failover error, or to continue in case of a successful failover.

[Table 11-8](#) lists the FailoverReturnCode enumeration values with a description of each enumerated value.

Table 11-8 FailoverReturnCode Enumeration Values

Member Names	Description
FailoverReturnCode.Retry	Requests ODP.NET to retry failover in case FailoverEvent.Error is passed to the application
FailoverReturnCode.Success	Requests ODP.NET to proceed so that the application receive more notifications, if any

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	Oracle.DataAccess.dll
Namespace	Oracle.DataAccess.Client
.NET Framework	See System Requirements

① See Also

- [FailoverEvent Enumeration](#)
- ["OracleFailoverEventArgs Class"](#)
- ["FailoverEvent"](#)
- *Oracle Real Application Clusters Administration and Deployment Guide*
- *Oracle AI Database Net Services Reference*

FailoverType Enumeration

FailoverType enumerated values are used to indicate the type of failover event that was raised.

[Table 11-9](#) lists all the FailoverType enumeration values with a description of each enumerated value.

Table 11-9 FailoverType Enumeration Values

Member Names	Description
FailoverType.Transactiona	Indicates that the user has requested transactional failover

Table 11-9 (Cont.) FailoverType Enumeration Values

Member Names	Description
FailoverType.Session	Indicates that the user has requested only session failover
FailoverType.Select	Indicates that the user has requested select and session failover
FailoverType.Auto	Indicates that the user has requested Transparent Application Continuity

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	Oracle.DataAccess.dll
Namespace	Oracle.DataAccess.Client
.NET Framework	See System Requirements

① See Also

- [FailoverEvent Enumeration](#)
- ["OracleFailoverEventArgs Class"](#)
- ["FailoverType"](#)
- *Oracle Real Application Clusters Administration and Deployment Guide*
- *Oracle AI Database Net Services Reference*

OracleFailoverEvent Enumeration

OracleFailoverEvent enumerated values are used to specify the state of the failover.

OracleFailoverEvent Enumeration Values lists all the OracleFailoverEvent enumeration values with a description of each enumerated value.

Table 11-10 OracleFailoverEvent Enumeration Values

Member Names	Description
OracleFailoverEvent.Begin	Indicates that failover has detected a lost connection and that failover is starting.
OracleFailoverEvent.End	Indicates successful completion of failover.
OracleFailoverEvent.Abort	Indicates that failover was unsuccessful, and there is no option of retrying.
OracleFailoverEvent.Error	Indicates that failover was unsuccessful, and it gives the application the opportunity to handle the error and retry failover. The application can retry failover by returning <code>OracleFailoverReturnCode.Retry</code> for the event notification.

Table 11-10 (Cont.) OracleFailoverEvent Enumeration Values

Member Names	Description
OracleFailoverEvent.Reauth	Indicates that a user handle has been reauthenticated. This applies to the situation where a client has multiple user sessions on a single server connection. During the initial failover, only the active user session is failed over. Other sessions are failed over when the application tries to use them. This is the value passed to the callback during these subsequent failovers.

No significant database operation should occur immediately after a OracleFailoverEvent.Begin event. SQL and major database operations should wait until the OracleFailoverEvent.End event. OracleFailoverEvent.Begin is primarily used to reject failover or to trace it. OracleFailoverEvent.Begin can also be used for non-database application operations, such as informing the end user a failover is in progress and to wait until it completes before proceeding. Transactions can be used in the OracleFailoverEvent.End callback phase, such as to file fault tickets or audit. These transactions must be committed before the callback completes.

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

See Also

- [OracleTAFMode Class](#)
- Oracle Real Application Clusters Administration and Deployment Guide*
- Oracle AI Database Net Services Reference*

OracleFailoverRestore Enumeration

OracleFailoverRestore enumerated values are used to indicates whether ODP.NET must restore session state after failover for Transparent Application Failover.

OracleFailoverRestore Enumeration Values lists all the OracleFailoverRestore enumeration values with a description of each enumerated value.

Table 11-11 OracleFailoverRestore Enumeration Values

Member Names	Description
OracleFailoverRestore.None	No session state will be restored after failover
OracleFailoverRestore.Level1	Basic session state will be restored after failover

Basic session state (Level 1) restored includes:

- Action
- Call collect time
- Client identifier
- Client info
- Container
- Current schema
- Database operation
- Edition
- Error on overlap time
- Execution context identifier
- Execution context sequence
- Module
- Row archival visibility
- Service
- SQL translation profile
- Time zone
- NLS parameters
 - NLS_LANGUAGE
 - NLS_TERRITORY
 - NLS_CURRENCY
 - NLS_ISO_CURRENCY
 - NLS_NUMERIC_CHARACTERS
 - NLS_CALENDAR
 - NLS_DATE_FORMAT
 - NLS_DATE_LANGUAGE
 - NLS_SORT
 - NLS_COMP
 - NLS_DUAL_CURRENCY
 - NLS_TIME_FORMAT

- NLS_TIMESTAMP_FORMAT
- NLS_TIME_TZ_FORMAT
- NLS_TIMESTAMP_TZ_FORMAT

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

 See Also

- [OracleTAFMode Class](#)
- *Oracle Real Application Clusters Administration and Deployment Guide*
- *Oracle AI Database Net Services Reference*

OracleFailoverReturnCode Enumeration

OracleFailoverReturnCode enumerated values are passed back by the application to the ODP.NET provider to request a retry in case of a failover error, or to continue in case of a successful failover.

OracleFailoverReturnCode Enumeration Values lists the OracleFailoverReturnCode enumeration values with a description of each enumerated value.

Table 11-12 OracleFailoverReturnCode Enumeration Values

Member Names	Description
OracleFailoverReturnCode.Retry	Requests ODP.NET to retry failover in case OracleFailoverEvent.Error is passed to the application
OracleFailoverReturnCode.Success	Requests ODP.NET to proceed so that the application receives more notifications, if any

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client

Provider	ODP.NET, Managed Driver	ODP.NET Core
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

 See Also

- [OracleTAFMode Class](#)
- *Oracle Real Application Clusters Administration and Deployment Guide*
- *Oracle AI Database Net Services Reference*

OracleFailoverType Enumeration

OracleFailoverType enumerated values are used to indicate the type of failover event that was raised.

OracleFailoverType Enumeration Values lists all the OracleFailoverType enumeration values with a description of each enumerated value.

Table 11-13 OracleFailoverType Enumeration Values

Member Names	Description
OracleFailoverType.None	Indicates the user has requested no failover functionality
OracleFailoverType.Session	Indicates the user has requested only session failover

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

 See Also

- [OracleTAFMode Class](#)
- *Oracle Real Application Clusters Administration and Deployment Guide*
- *Oracle AI Database Net Services Reference*

12

Oracle Database Advanced Queuing and Transactional Event Queues Classes

This chapter describes the following Oracle Data Provider for .NET classes:

- [OracleAQAgent Class](#)
- [OracleAQDequeueOptions Class](#)
- [OracleAQEnqueueOptions Class](#)
- [OracleAQMessage Class](#)
- [OracleAQMessageAvailableEventArgs Class](#)
- [OracleAQMessageAvailableEventHandler Delegate](#)
- [OracleAQQueue Class](#)
- [OracleAQDequeueMode Enumeration](#)
- [OracleAQMessageDeliveryMode Enumeration](#)
- [OracleAQMessageState Enumeration](#)
- [OracleAQMessageType Enumeration](#)
- [OracleAQNavigationMode Enumeration](#)
- [OracleAQNotificationGroupingType Enumeration](#)
- [OracleAQNotificationType Enumeration](#)
- [OracleAQVisibilityMode Enumeration](#)
- [OracleTxEventQNotificationDirective Enumeration](#)

OracleAQAgent Class

The `OracleAQAgent` class represents agents that may be senders or recipients of a message.

Class Inheritance

`System.Object`

`OracleAQAgent`

Declaration

```
// C#  
public sealed class OracleAQAgent
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

An agent may be a consumer, another queue, or a consumer of another queue. The queue may be either local or remote. A remote queue is specified through a database link.

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleAQAgent Members](#)
- [OracleAQAgent Constructors](#)
- [OracleAQAgent Properties](#)

OracleAQAgent Members

OracleAQAgent members are listed in the following tables.

OracleAQAgent Constructors

OracleAQAgent constructors are listed in [Table 12-1](#).

Table 12-1 OracleAQAgent Constructors

Constructor	Description
OracleAQAgent Constructors	Instantiates a new instance of the OracleAQAgent class (Overloaded).

OracleAQAgent Properties

OracleAQAgent properties are listed in [Table 12-2](#).

Table 12-2 OracleAQAgent Properties

Property	Description
Address	Specifies the address of the agent.
Name	Specifies the name of the agent.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleAQAgent Class](#)

OracleAQAgent Constructors

OracleAQAgent constructors instantiate new instances of the OracleAQAgent class.

Overload List:

- [OracleAQAgent \(string\)](#)
This constructor instantiates the OracleAQAgent class using the specified name.
- [OracleAQAgent \(string, string\)](#)
This constructor instantiates the OracleAQAgent class using the specified name and address.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleAQAgent Class](#)
- [OracleAQAgent Members](#)

OracleAQAgent (string)

This constructor instantiates the OracleAQAgent class using the specified name.

Declaration

```
// C#  
public OracleAQAgent(string name);
```

Parameters

- *name*
The name of the agent.

Exceptions

ArgumentNullException - The *name* parameter is null.

ArgumentException - The *name* parameter is empty.

Remarks

The agent name signifies the name of a producer or consumer of a message. In the context of functionality exposed by `Listen`, an agent name corresponds to the name of a consumer for which a message is expected on a multiconsumer queue. It may also be set on a message to signify sender identification or intended recipients of the message.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQAgent Class](#)
- [OracleAQAgent Members](#)

OracleAQAgent (string, string)

This constructor instantiates the `OracleAQAgent` class using the specified name and address.

Declaration

```
// C#  
public OracleAQAgent(string name, string address);
```

Parameters

- *name*
The name of the agent.
- *address*
The address is of the form `[schema].queue[@dbLink]`.

Exceptions

ArgumentNullException - The *address* parameter is null.

ArgumentException - The *address* parameter is empty.

Remarks

The agent name signifies the name of a producer or consumer of a message. In the context of functionality exposed by `Listen`, an agent name corresponds to the name of a consumer for which a message is expected on a multiconsumer queue. For multiconsumer queue, the agent name must be a non null value.

The *name* parameter can be specified as `null` in this constructor. In such a scenario, the agent only has an *address*.

The *address* parameter signifies the name of the queue against which this agent listens for new messages. The *address* represents a queue at a local or remote database. The validity of

the *address* is not checked implicitly. The exceptions due to wrong *address* are thrown only during database operations such as Listen.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQAgent Class](#)
- [OracleAQAgent Members](#)

OracleAQAgent Properties

OracleAQAgent properties are listed in [Table 12-3](#).

Table 12-3 OracleAQAgent Properties

Property	Description
Address	Specifies the address of the agent.
Name	Specifies the name of the agent.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQAgent Class](#)
- [OracleAQAgent Members](#)

Address

This instance property specifies the address of the agent.

Declaration

```
// C#  
public string Address {get; }
```

Property Value

A string that specifies the agent address.

Remarks

The address represents a queue at a local or remote database. The default value is null. The address of the agent is of the form *[schema.]queue[dbLink]*. The string length can be up to 1024 characters.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQAgent Class](#)
- [OracleAQAgent Members](#)

Name

This instance property specifies the name of the agent.

Declaration

```
// C#  
public string Name {get; }
```

Property Value

A string.

Remarks

The default is null. The string length can be up to 128 characters. A non-null value implies that this agent name either corresponds to a consumer name in a multiconsumer queue, or a recipient as specified in message properties.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQAgent Class](#)
- [OracleAQAgent Members](#)

OracleAQDequeueOptions Class

An OracleAQDequeueOptions object represents the options available when dequeuing a message from an OracleAQQueue object.

Class Inheritance

System.Object

OracleAQDequeueOptions

Declaration

```
// C#  
public sealed class OracleAQDequeueOptions : ICloneable
```


Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQDequeueOptions Members](#)
- [OracleAQDequeueOptions Constructor](#)
- [OracleAQDequeueOptions Properties](#)
- [OracleAQDequeueOptions Public Methods](#)

OracleAQDequeueOptions Members

OracleAQDequeueOptions members are listed in the following tables.

OracleAQDequeueOptions Constructor

The OracleAQDequeueOptions constructor is listed in [Table 12-4](#).

Table 12-4 OracleAQDequeueOptions Constructor

Constructor	Description
OracleAQDequeueOptions Constructor	Instantiates a new instance of the OracleAQDequeueOptions class

OracleAQDequeueOptions Properties

OracleAQDequeueOptions properties are listed in [Table 12-5](#).

Table 12-5 OracleAQDequeueOptions Properties

Property	Description
ConsumerName	Specifies the consumer name for which to dequeue the message
Correlation	Specifies the correlation identifier of the message to be dequeued
DeliveryMode	Specifies the expected delivery mode of the message being dequeued
DequeueMode	Specifies the locking behavior associated with the dequeue operation
MessageId	Specifies the message identifier of the message to be dequeued
NavigationMode	Specifies the position of the message that will be retrieved
ProviderSpecificType	Specifies whether the payload of a dequeued message is provided as an ODP.NET specific type or a .NET type
Visibility	Specifies whether or not the new message is dequeued as part of the current transaction
Wait	Specifies the wait time, in seconds, for a message that matches the search criteria

OracleAQDequeueOptions Public Methods

OracleAQDequeueOptions public methods are listed in [Table 12-6](#).

Table 12-6 OracleAQDequeueOptions Public Methods

Public Method	Description
Clone	Creates a copy of an OracleAQDequeueOptions object.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleAQDequeueOptions Class](#)

OracleAQDequeueOptions Constructor

The OracleAQDequeueOptions constructor creates an instance of the OracleAQDequeueOptions class and sets all its properties to their default values.

Declaration

```
// C#  
public OracleAQDequeueOptions();
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQDequeueOptions Class](#)
- [OracleAQDequeueOptions Members](#)

OracleAQDequeueOptions Properties

OracleAQDequeueOptions properties are listed in [Table 12-7](#).

Table 12-7 OracleAQDequeueOptions Properties

Property	Description
ConsumerName	Specifies the consumer name for which to dequeue the message
Correlation	Specifies the correlation identifier of the message to be dequeued
DeliveryMode	Specifies the expected delivery mode of the message being dequeued
DequeueMode	Specifies the locking behavior associated with the dequeue operation
MessageId	Specifies the message identifier of the message to be dequeued
NavigationMode	Specifies the position of the message that will be retrieved
ProviderSpecificType	Specifies whether the payload of a dequeued message is provided as an ODP.NET specific type or a .NET type
Visibility	Specifies whether or not the new message is dequeued as part of the current transaction
Wait	Specifies the wait time, in seconds, for a message that matches the search criteria

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQDequeueOptions Class](#)
- [OracleAQDequeueOptions Members](#)

ConsumerName

This instance property specifies the consumer name for which to dequeue the message.

Declaration

```
// C#  
public string ConsumerName {get;set;}
```

Property Value

A string.

Remarks

The ConsumerName property only accesses those messages that match the consumer name. If a queue is not set up for multiple consumers, then this field should be set to null.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQDequeueOptions Class](#)
- [OracleAQDequeueOptions Members](#)

Correlation

This instance property specifies the correlation identifier of the message to be dequeued.

Declaration

```
// C#  
public string Correlation {get;set;}
```

Property Value

A string.

Remarks

This property specifies the identification of the message to be dequeued. Special pattern matching characters, such as the percent sign (%) and the underscore (_) can be used. If more than one message satisfies the pattern, then the order of dequeuing is undetermined.

The maximum length of Correlation is 128.

MessageId and Correlation are two independent identifiers. While MessageId is unique for a message, a group of messages can be assigned the same Correlation. Also, pattern matching is possible only with Correlation.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQDequeueOptions Class](#)
- [OracleAQDequeueOptions Members](#)

DeliveryMode

This instance property specifies the expected delivery mode of the message being dequeued.

Declaration

```
// C#  
public OracleAQMessageDeliveryMode DeliveryMode {get;set;}
```

Property Value

An `OracleAQMessageDeliveryMode` enumerated value.

Remarks

This property specifies the type of messages to be dequeued. It can be set to dequeue either persistent or buffered messages, or both from a queue. The following values are valid:

- `OracleAQMessageDeliveryMode.Persistent`
- `OracleAQMessageDeliveryMode.Buffered`
- `OracleAQMessageDeliveryMode.PersistentOrBuffered`

The default value is `OracleAQMessageDeliveryMode.Persistent`.

`OracleAQMessageDeliveryMode.Buffered` is not supported for AQ and TxEventQ operations, the workaround is to use `OracleAQMessageDeliveryMode.Persistent`.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQDequeueOptions Class](#)
- [OracleAQDequeueOptions Members](#)

DequeueMode

This instance property specifies the locking behavior associated with the dequeue operation.

Declaration

```
// C#  
public OracleAQDequeueMode DequeueMode {get;set;}
```

Property Value

An `OracleAQDequeueMode` enumerated value.

Exceptions

`ArgumentOutOfRangeException` - The specified `DequeueMode` value is invalid.

Remarks

The default value is `OracleAQDequeueMode.Remove`.

`OracleAQDequeueMode.RemoveNoData` is not supported for TxEventQ . In case of AQ, it is not supported with JSON data type.

OracleAQDequeueMode.Locked is not supported with Visibility = Immediate for AQ and TxEventQ, the workaround is to use OracleAQDequeueMode.Locked with Visibility = OnCommit.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQDequeueOptions Class](#)
- [OracleAQDequeueOptions Members](#)

MessageId

This instance property specifies the message identifier of the message to be dequeued.

Declaration

```
// C#  
public byte[] MessageId {get;set;}
```

Property Value

A byte[].

Remarks

The dequeue operation succeeds only if the message ID of the message being dequeued matches with the message ID specified.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQDequeueOptions Class](#)
- [OracleAQDequeueOptions Members](#)

NavigationMode

This instance property specifies the position of the message that will be retrieved.

Declaration

```
// C#  
public OracleAQNavigationMode NavigationMode {get;set;}
```

Property Value

An OracleAQNavigationMode enumerated value.

Exceptions

ArgumentOutOfRangeException - The specified NavigationMode value is invalid.

Remarks

The default value is `OracleAQNavigationMode.NextMessage`.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQDequeueOptions Class](#)
- [OracleAQDequeueOptions Members](#)

ProviderSpecificType

This property specifies whether the payload of a dequeued message is provided as an ODP.NET specific type or a .NET type.

Declaration

```
// C#  
public bool ProviderSpecificType {get;set;}
```

Property Value

A `bool`.

Remarks

The default value of this property is `false`. For a discussion of how this property affects payload type, refer to ["MessageType"](#) under the `OracleAQQueue` class.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQDequeueOptions Class](#)
- [OracleAQDequeueOptions Members](#)
- ["MessageType"](#)

Visibility

This instance property specifies whether or not the new message is dequeued as part of the current transaction.

Declaration

```
// C#  
public OracleAQVisibilityMode Visibility {get;set;}
```

Property Value

An `OracleAQVisibilityMode` enumerated value.

Exceptions

`ArgumentOutOfRangeException` - The `Visibility` value specified is invalid.

Remarks

The default value is `OracleAQVisibilityMode.OnCommit`. You must use transactions when using the default value for this property. This ensures that applications do not lose messages and the messages are appropriately removed from the queue after the dequeue operation is successful. If transactions are not used when using the default visibility mode of `OracleAQVisibilityMode.OnCommit`, then messages are not removed from the queue.

Using the alternative visibility mode value, `OracleAQVisibilityMode.Immediate` can eliminate the need to create, commit, and rollback a transaction. However, if an error occurs during the dequeue operation, then the message may be lost.

The visibility parameter is ignored when `DequeueMode` is set to `OracleAQDequeueMode.Browse`.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQDequeueOptions Class](#)
- [OracleAQDequeueOptions Members](#)

Wait

This instance property specifies the wait time, in seconds, for a message that matches the search criteria.

Declaration

```
// C#  
public int Wait {get;set;}
```

Property Value

Any positive integer value or 0 or -1.

Exceptions

`ArgumentOutOfRangeException` - The specified `Wait` value is invalid.

Remarks

The default value is -1, which implies an infinite wait. The following values are valid:

- Positive integer: Wait time in seconds.
- -1: Wait forever.
- 0: Do not wait.

A value of less than -1 raises an `ArgumentOutOfRangeException`.

This parameter is ignored if messages in the same group are being dequeued.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQDequeueOptions Class](#)
- [OracleAQDequeueOptions Members](#)

OracleAQDequeueOptions Public Methods

The OracleAQDequeueOptions public method is listed in [Table 12-8](#).

Table 12-8 OracleAQDequeueOptions Public Methods

Public Method	Description
Clone	Creates a copy of an OracleAQDequeueOptions object

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQDequeueOptions Class](#)
- [OracleAQDequeueOptions Members](#)

Clone

This method creates a copy of an OracleAQDequeueOptions object.

Declaration

```
// C#
public object Clone();
```

Return Value

An OracleAQDequeueOptions object.

Implements

ICloneable.

Remarks

The cloned object has the same property values as the object being cloned.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQDequeueOptions Class](#)
- [OracleAQDequeueOptions Members](#)

OracleAQEnqueueOptions Class

The OracleAQEnqueueOptions class represents the options available when enqueueing a message to an OracleAQQueue.

Class Inheritance

System.Object
 OracleAQEnqueueOptions

Declaration

```
// C#  
public sealed class OracleAQEnqueueOptions : ICloneable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQEnqueueOptions Members](#)
- [OracleAQEnqueueOptions Constructor](#)
- [OracleAQEnqueueOptions Properties](#)
- [OracleAQEnqueueOptions Public Methods](#)

OracleAQEnqueueOptions Members

The OracleAQEnqueueOptions members are listed in the following tables.

OracleAQEnqueueOptions Constructor

OracleAQEnqueueOptions constructor is listed in [Table 12-9](#).

Table 12-9 OracleAQEnqueueOptions Constructor

Constructor	Description
OracleAQEnqueueOptions Constructor	Instantiates a new instance of the OracleAQEnqueueOptions class.

OracleAQEnqueueOptions Properties

OracleAQEnqueueOptions properties are listed in [Table 12-10](#).

Table 12-10 OracleAQEnqueueOptions Properties

Property	Description
DeliveryMode	Specifies the delivery mode of the message being enqueued.
Visibility	Specifies whether or not the new message is enqueued as part of the current transaction.

OracleAQEnqueueOptions Public Methods

The OracleAQEnqueueOptions public method is listed in [Table 12-11](#).

Table 12-11 OracleAQEnqueueOptions Public Methods

Public Method	Description
Clone	Creates a copy of an OracleAQEnqueueOptions object.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQEnqueueOptions Class](#)

OracleAQEnqueueOptions Constructor

This constructor creates an instance of the OracleAQEnqueueOptions class with default property values.

Declaration

```
// C#  
public OracleAQEnqueueOptions();
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQEnqueueOptions Class](#)
- [OracleAQEnqueueOptions Members](#)

OracleAQEnqueueOptions Properties

OracleAQEnqueueOptions properties are listed in [Table 12-12](#).

Table 12-12 OracleAQEnqueueOptions Properties

Property	Description
DeliveryMode	Specifies the delivery mode of the message being enqueued.
Visibility	Specifies whether or not the new message is enqueued as part of the current transaction.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQEnqueueOptions Class](#)
- [OracleAQEnqueueOptions Members](#)

DeliveryMode

This instance property specifies the delivery mode of the message being enqueued.

Declaration

```
// C#  
public OracleAQMessageDeliveryMode DeliveryMode {get;set;}
```

Exceptions

ArgumentOutOfRangeException - The specified Visibility value is invalid.

Remarks

The valid values can be any of the following enumerated values:

- `OracleAQMessageDeliveryMode.Persistent`
- `OracleAQMessageDeliveryMode.Buffered`

The default is `OracleAQMessageDeliveryMode.Persistent`.

`OracleAQMessageDeliveryMode.PersistentOrBuffered` cannot be set on this property.

OracleAQMessageDeliveryMode.Buffered is not supported for AQ and TxEventQ operations, workaround is to use OracleAQMessageDeliveryMode.Persistent.

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleAQEnqueueOptions Class](#)
- [OracleAQEnqueueOptions Members](#)

Visibility

This instance property specifies whether or not the new message is enqueued as part of the current transaction.

Declaration

```
// C#  
public OracleAQVisibilityMode Visibility {get;set;}
```

Property Value

An OracleAQVisibilityMode enumerated value.

Exceptions

ArgumentOutOfRangeException - The specified Visibility value is invalid.

Remarks

The default value is OracleAQVisibilityMode.OnCommit. You must use transactions when using the default value. If transactions are not used when using the default visibility mode of OracleAQVisibilityMode.OnCommit, then messages are not enqueued to the queue.

Using the alternative visibility mode value, OracleAQVisibilityMode.Immediate eliminates the need to use a transaction. The queue is not affected in case the enqueue operation fails. The message does not get enqueued to the queue for such cases.

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleAQEnqueueOptions Class](#)
- [OracleAQEnqueueOptions Members](#)

OracleAQEnqueueOptions Public Methods

OracleAQEnqueueOptions public method is listed in [Table 12-13](#).

Table 12-13 OracleAQEnqueueOptions Public Methods

Public Method	Description
Clone	Creates a copy of an OracleAQEnqueueOptions object.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQEnqueueOptions Class](#)
- [OracleAQEnqueueOptions Members](#)

Clone

This method creates a copy of an OracleAQEnqueueOptions object.

Declaration

```
// C#
public object Clone();
```

Return Value

An OracleAQEnqueueOptions object.

Implements

ICloneable.

Remarks

The cloned object has the same property values as that of the object being cloned.

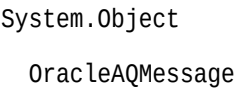
See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQEnqueueOptions Class](#)
- [OracleAQEnqueueOptions Members](#)

OracleAQMessage Class

An OracleAQMessage object represents a message to be enqueued and dequeued.

Class Inheritance



Declaration

```
// C#
public sealed class OracleAQMessage
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

An OracleAQMessage object consists of control information (metadata) and Payload (data). The control information is exposed by various properties on the OracleAQMessage object and is used by Oracle Database Advanced Queuing to manage messages. The payload is the information stored in the queue.

❗ Note

An instance of OracleAQMessage cannot be re-used across multiple operations of OracleAQQueue public method [Enqueue\(\)](#) or [EnqueueArray\(\)](#), if the payload is an XmlReader. This is a direct consequence of the forward-only semantics of the XmlReader, as an [Enqueue\(\)](#) or [EnqueueArray\(\)](#) operation internally invokes a read operation on the XmlReader to extract the data to be enqueued.

❗ See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessage Members](#)
- [OracleAQMessage Constructors](#)
- [OracleAQMessage Properties](#)

OracleAQMessage Members

OracleAQMessage members are listed in the following tables.

OracleAQMessage Constructor

OracleAQMessage constructors are listed in [Table 12-14](#).

Table 12-14 OracleAQMessage Constructors

Constructor	Description
OracleAQMessage Constructors	Instantiates a new instance of the OracleAQMessage class (Overloaded).

OracleAQMessage Properties

OracleAQMessage properties are listed in [Table 12-15](#).

Table 12-15 OracleAQMessage Properties

Property	Description
Correlation	Specifies an identification for the message.
Delay	Specifies the duration, in seconds, after which an enqueued message is available for dequeuing.
DeliveryMode	Specifies the delivery mode of the dequeued message.
DequeueAttempts	Returns the number of attempts that have been made to dequeue the message.
EnqueueTime	Specifies the time when the message was enqueued.
ExceptionQueue	Specifies the name of the queue that the message should be moved to if it cannot be processed successfully.
Expiration	Specifies the duration, in seconds, for which an enqueued message is available for dequeuing.
MessageId	Returns the message identifier.
OriginalMessageId	Specifies the identifier of the message in the last queue that generated this message.
Payload	Specifies the payload of the message.
Priority	Specifies the priority of the message.
Recipients	Specifies the list of recipients that overrides the default queue subscribers.
SenderId	Identifies the original sender of the message.
State	Specifies the state of the message at the time of dequeue.
TransactionGroup	Specifies the transaction group for the dequeued message.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessage Class](#)

OracleAQMessage Constructors

OracleAQMessage constructors create new instances of the OracleAQMessage class.

Overload List:

- [OracleAQMessage\(\)](#)
This constructor instantiates the OracleAQMessage class.
- [OracleAQMessage\(Object\)](#)
This constructor instantiates the OracleAQMessage class using the object provided as the payload.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessage Class](#)
- [OracleAQMessage Members](#)

OracleAQMessage()

This constructor instantiates the OracleAQMessage class.

Declaration

```
// C#  
public OracleAQMessage();
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessage Class](#)
- [OracleAQMessage Members](#)

OracleAQMessage(Object)

This constructor instantiates the OracleAQMessage class using the Object provided as the *payload*.

Declaration

```
// C#  
public OracleAQMessage(Object payload);
```

Parameters

- *payload*

An Object specifying *payload*. It can be one of the following types:

- byte[]
- IOracleCustomType
- OracleBinary
- OracleString
- OracleXmlType
- String
- XmlReader

Exceptions

ArgumentException - The specified *payload* value is of invalid type.

Remarks

The ODP.NET AQ implementation currently does not support user defined types with LOB attributes. It also does not support other variants of user defined types such as VARRAY and nested tables, as Oracle Database AQ does not support them inherently.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessage Class](#)
- [OracleAQMessage Members](#)

OracleAQMessage Properties

OracleAQMessage properties are listed in [Table 12-16](#).

Table 12-16 OracleAQMessage Properties

Property	Description
Correlation	Specifies an identification for the message.
Delay	Specifies the duration, in seconds, after which an enqueued message is available for dequeuing.
DeliveryMode	Specifies the delivery mode of the dequeued message.
DequeueAttempts	Returns the number of attempts that have been made to dequeue the message.

Table 12-16 (Cont.) OracleAQMessage Properties

Property	Description
EnqueueTime	Specifies the time when the message was enqueued.
ExceptionQueue	Specifies the name of the queue that the message should be moved to if it cannot be processed successfully.
Expiration	Specifies the duration, in seconds, for which an enqueued message is available for dequeuing.
MessageId	Returns the message identifier.
OriginalMessageId	Specifies the identifier of the message in the last queue that generated this message.
Payload	Specifies the payload of the message.
Priority	Specifies the priority of the message.
Recipients	Specifies the list of recipients that overrides the default queue subscribers.
SenderId	Identifies the original sender of the message.
State	Specifies the state of the message at the time of dequeue.
TransactionGroup	Specifies the transaction group for the dequeued message.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessage Class](#)
- [OracleAQMessage Members](#)

Correlation

This instance property specifies an identification for the message.

Declaration

```
// C#  
public string Correlation {get;set;}
```

Property Value

A string that specifies the identification for the message.

Remarks

The producer of a message can set this property at the time of enqueueing. The consumer can then use this identification to dequeue specific messages by setting the Correlation property of an OracleAQDequeueOptions object. For more information regarding dequeuing messages based on Correlation, refer to ["Correlation"](#) under the OracleAQDequeueOptions class.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessage Class](#)
- [OracleAQMessage Members](#)
- ["Correlation"](#)

Delay

This instance property specifies the duration, in seconds, after which an enqueued message is available for dequeuing.

Declaration

```
// C#  
public int Delay {get;set;}
```

Property Value

An integer that indicates the number of seconds after which an enqueued message is available for dequeuing.

Exceptions

ArgumentException - The value specified is less than 0.

Remarks

This property delays the immediate consumption of an enqueued message. The following are valid values for this property:

- Positive integer - Indicates the delay in seconds.
- 0 - indicates that the message is immediately available for dequeuing.

The default value is 0. The Delay property is not supported with buffered messaging.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessage Class](#)
- [OracleAQMessage Members](#)

DeliveryMode

This instance property specifies the delivery mode of the dequeued message.

Declaration

```
// C#  
public OracleAQMessageDeliveryMode DeliveryMode {get;}
```

Property Value

An `OracleAQMessageDeliveryMode` enumerated value
(`OracleAQMessageDeliveryMode.Persistent` or `OracleAQMessageDeliveryMode.Buffered`).

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessage Class](#)
- [OracleAQMessage Members](#)

DequeueAttempts

This instance property returns the number of attempts that have been made to dequeue the message.

Declaration

```
// C#  
public int DequeueAttempts {get;}
```

Property Value

An integer that indicates the number of dequeue attempts.

Remarks

This property is available in an `OracleAQMessage` after the message has been dequeued from a queue.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessage Class](#)
- [OracleAQMessage Members](#)

EnqueueTime

This instance property specifies the time when the message was enqueued.

Declaration

```
// C#  
public DateTime EnqueueTime {get;}
```

Property Value

A `DateTime` object.

Remarks

This property is available after the message is dequeued. It provides the enqueue time of a dequeued message. The time granularity is in seconds as that is the granularity the database provides.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessage Class](#)
- [OracleAQMessage Members](#)

ExceptionQueue

This instance property specifies the name of the queue that the message should be moved to if it cannot be processed successfully.

Declaration

```
// C#  
public string ExceptionQueue {get;set;}
```

Property Value

The name of the queue that a message should be moved to if it cannot be processed successfully. The default value is null.

Remarks

This property specifies the queue that a message should be moved to if the message has expired or if the number of unsuccessful dequeue attempts have exceeded the `max_retries` value for the queue.

If this property is not set or the specified exception queue name does not exist, then the default exception queue associated with the queue table is used.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessage Class](#)
- [OracleAQMessage Members](#)

Expiration

This instance property specifies the duration, in seconds, for which an enqueued message is available for dequeuing.

Declaration

```
// C#  
public int Expiration {get;set;}
```

Property Value

An integer that specifies the number of seconds an enqueued message is available for dequeuing.

Exceptions

ArgumentException - The value specified is less than -1.

Remarks

The value specified is an offset from the value specified in the Delay property.

The following are valid values for the property:

- Positive integer - Indicates the expiration in seconds.
- -1 - Indicates that the message never expires.

The default value is -1. When a message expires, the message moves from the READY state to the EXPIRED state.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessage Class](#)
- [OracleAQMessage Members](#)

MessageId

This instance property returns the message identifier.

Declaration

```
// C#  
public byte[] MessageId {get;}
```

Property Value

A byte[] that specifies the message identifier.

Remarks

This property is available after an enqueue or dequeue operation. Dequeued buffered messages have a null value for MessageId.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessage Class](#)
- [OracleAQMessage Members](#)

OriginalMessageId

This instance property specifies the identifier of the message in the last queue that generated this message.

Declaration

```
// C#  
public byte[] OriginalMessageId {get;}
```

Property Value

A byte[] that specifies the original message identifier.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessage Class](#)
- [OracleAQMessage Members](#)

Payload

This instance property specifies the payload of the message.

Declaration

```
// C#  
public Object Payload {get;set;}
```

Property Value

An Object that specifies the payload of the message.

Exceptions

ArgumentException - The specified object is not one of the allowed types.

Remarks

For a complete discussion of various payload types, refer to "[MessageType](#)" under the OracleAQQueue class.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessage Class](#)
- [OracleAQMessage Members](#)
- ["MessageType"](#)

Priority

This instance property specifies the priority of the message.

Declaration

```
// C#  
public int Priority {get;set;}
```

Property Value

An integer that specifies the priority of the message.

Remarks

The default value is 0. In order to take effect, this property must be set prior to enqueueing the message.

Smaller values indicate higher priority for the message. Negative values may also be used.

The priority of an enqueued message is useful for priority-based dequeuing.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessage Class](#)
- [OracleAQMessage Members](#)

Recipients

This instance property specifies the list of recipients that overrides the default queue subscribers.

Declaration

```
// C#  
public OracleAQAgent[] Recipients {get; set;}
```

Property Value

An `OracleAQAgent[]`.

Remarks

This recipient list is valid only for messages being enqueued to multiconsumer queues. The list of recipients is not returned with the message at the time of dequeuing.

TxEvtQ does not support recipient list.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessage Class](#)
- [OracleAQMessage Members](#)

SenderId

This instance property identifies the original sender of the message.

Declaration

```
// C#  
public OracleAQAgent SenderId {get; set;}
```

Property Value

An `OracleAQAgent` object.

Remarks

Sender identification is supported in all queue tables.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessage Class](#)
- [OracleAQMessage Members](#)

State

This instance property specifies the state of the message at the time of dequeue.

Declaration

```
// C#  
public OracleAQMessageState State {get;}
```

Property Value

An `OracleAQMessageState` enumerated value.

Remarks

This property is available after the message is dequeued.

The state of buffered messages dequeued by specifying Correlation under dequeue options is always `OracleAQMessageState.Ready`.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessage Class](#)
- [OracleAQMessage Members](#)

TransactionGroup

This instance property specifies the transaction group for the dequeued message.

Declaration

```
// C#  
public string TransactionGroup {get;}
```

Property Value

A string that specifies the transaction group.

Remarks

Messages belonging to one queue can be grouped to form a set that can only be consumed by one user at a time. This requires that the queue be created in a queue table that is enabled for message grouping. All messages belonging to a group must be created in the same transaction. Also, all messages created in one transaction belong to the same group.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessage Class](#)
- [OracleAQMessage Members](#)

OracleAQMessageAvailableEventArgs Class

The `OracleAQMessageAvailableEventArgs` class provides event data for the `OracleAQQueue.MessageAvailable` event.

Class Inheritance

`System.Object`

`System.EventArgs`

Oracle.DataAccess.Client.OracleAQMessageAvailableEventArgs

Declaration

```
// C#  
public sealed class OracleAQMessageAvailableEventArgs
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

This class cannot be inherited.

For detailed information on all the inherited properties and methods, please read the documentation provided by Microsoft's .NET Documentation.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessageAvailableEventArgs Members](#)
- [OracleAQMessageAvailableEventArgs Constructor](#)
- [OracleAQMessageAvailableEventArgs Properties](#)

OracleAQMessageAvailableEventArgs Members

OracleAQMessageAvailableEventArgs members are listed in the following tables.

OracleAQMessageAvailableEventArgs Constructor

OracleAQMessageAvailableEventArgs constructor is listed in [Table 12-17](#)

Table 12-17 OracleAQMessageAvailableEventArgs Constructor

Property	Description
OracleAQMessageAvailableEventArgs Constructor	Instantiates a new instance of the OracleAQMessageAvailableEventArgs class.

OracleAQMessageAvailableEventArgs Properties

OracleAQMessageAvailableEventArgs properties are listed in [Table 12-18](#).

Table 12-18 OracleAQMessageAvailableEventArgs Properties

Property	Description
AvailableMessages	Specifies the number of messages that raised this notification.
ConsumerName	Provides the name of the consumer for which the message is available for dequeuing.
Correlation	Provides the user-defined identifier of the message.
Delay	Specifies the duration, in seconds, after which an enqueued message is available for dequeuing.
DeliveryMode	Specifies the delivery mode of the message.
EnqueueTime	Specifies the time when the message was enqueued.
ExceptionQueue	Specifies the name of the queue that the message is moved to if it cannot be processed successfully.
Expiration	Specifies the duration, in seconds, for which an enqueued message is available for dequeuing before expiring.
MessageId	Returns an array of message identifiers.
NotificationType	Indicates the type of notification such as regular, grouping, or timeout.
OriginalMessageId	Specifies the ID of the message, in the last queue, that generated this message.
Payload	Provides payload associated with the notification.
Priority	Specifies the priority of the message.
Queue	Indicates OracleAQQueue instance corresponding to which the notification is received.
QueueName	Indicates the name of the queue that contains the message to be dequeued.
SenderId	Identifies the original sender of the message.
State	Specifies the state of the message.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessageAvailableEventArgs Class](#)

OracleAQMessageAvailableEventArgs Constructor

This constructor creates an instance of the `OracleAQMessageAvailableEventArgs` class with default property values.

Declaration

```
// C#
public OracleAQMessageAvailableEventArgs();
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessageAvailableEventArgs Class](#)
- [OracleAQMessageAvailableEventArgs Members](#)

OracleAQMessageAvailableEventArgs Properties

`OracleAQMessageAvailableEventArgs` properties are listed in [Table 12-19](#).

Table 12-19 OracleAQMessageAvailableEventArgs Properties

Property	Description
AvailableMessages	Specifies the number of messages that raised this notification.
ConsumerName	Provides the name of the consumer for which the message is available for dequeuing.
Correlation	Provides the user-defined identifier of the message.
Delay	Specifies the duration, in seconds, after which an enqueued message is available for dequeuing.
DeliveryMode	Specifies the delivery mode of the message.
EnqueueTime	Specifies the time when the message was enqueued.
ExceptionQueue	Specifies the name of the queue that the message is moved to if it cannot be processed successfully.
Expiration	Specifies the duration, in seconds, for which an enqueued message is available for dequeuing before expiring.
MessageId	Returns an array of message identifiers.
NotificationType	Indicates the type of notification such as regular, grouping, or timeout.
OriginalMessageId	Specifies the ID of the message, in the last queue, that generated this message.
Payload	Provides payload associated with the notification.
Priority	Specifies the priority of the message.
Queue	Indicates <code>OracleAQQueue</code> instance corresponding to which the notification is received.

Table 12-19 (Cont.) OracleAQMessageAvailableEventArgs Properties

Property	Description
QueueName	Indicates the name of the queue that contains the message to be dequeued.
SenderId	Identifies the original sender of the message.
State	Specifies the state of the message.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessageAvailableEventArgs Class](#)
- [OracleAQMessageAvailableEventArgs Members](#)

AvailableMessages

This instance property specifies the number of messages that raised this notification.

Declaration

```
// C#  
public int AvailableMessages{get;}
```

Property Value

An integer indicating the number of messages that raised this notification.

Remarks

The property value is 1 for a regular notification type. The notification type can be specified using the `OracleAQQueue.Notification` property.

This property is not relevant if the `NotificationType` is `OracleAQNotificationType.Timeout`.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessageAvailableEventArgs Class](#)
- [OracleAQMessageAvailableEventArgs Members](#)

ConsumerName

This property provides the name of the consumer for which the message is available for dequeuing.

Declaration

```
// C#  
public string ConsumerName {get;}
```

Property Value

A string that identifies the name of the consumer.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessageAvailableEventArgs Class](#)
- [OracleAQMessageAvailableEventArgs Members](#)

Correlation

This instance property specifies the identification for the message.

Declaration

```
// C#  
public string Correlation {get;}
```

Property Value

A string that specifies the identification for the message.

Remarks

This property specifies the correlation of the message for which the notification is raised. The consumer can then use this identification to dequeue specific messages by setting the "[Correlation](#)" property of the `OracleAQDequeueOptions` object.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessageAvailableEventArgs Class](#)
- [OracleAQMessageAvailableEventArgs Members](#)
- ["Correlation"](#)

Delay

This instance property specifies the duration, in seconds, after which an enqueued message is available for dequeuing.

Declaration

```
// C#  
public int Delay {get;}
```

Property Value

An integer that indicates the duration, in seconds, after which an enqueued message is available for dequeuing.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessageAvailableEventArgs Class](#)
- [OracleAQMessageAvailableEventArgs Members](#)

DeliveryMode

This instance property specifies the delivery mode of the message.

Declaration

```
// C#  
public OracleAQMessageDeliveryMode DeliveryMode {get;}
```

Property Value

An OracleAQMessageDeliveryMode enumerated value.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessageAvailableEventArgs Class](#)
- [OracleAQMessageAvailableEventArgs Members](#)

EnqueueTime

This instance property specifies the time when the message was enqueued.

Declaration

```
// C#  
public DateTime EnqueueTime {get;}
```

Property Value

A DateTime object.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessageAvailableEventArgs Class](#)
- [OracleAQMessageAvailableEventArgs Members](#)

ExceptionQueue

This instance property specifies the name of the queue that the message is moved to if it cannot be processed successfully.

Declaration

```
// C#  
public string ExceptionQueue {get;}
```

Property Value

The name of the queue that a message to is moved if it cannot be processed successfully.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessageAvailableEventArgs Class](#)
- [OracleAQMessageAvailableEventArgs Members](#)

Expiration

This instance property specifies the duration, in seconds, for which an enqueued message is available for dequeuing before expiring.

Declaration

```
// C#  
public int Expiration {get;}
```

Property Value

An integer that specifies the duration, in seconds, for which an enqueued message is available for dequeuing.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessageAvailableEventArgs Class](#)
- [OracleAQMessageAvailableEventArgs Members](#)

MessageId

This instance property returns an array of message identifiers.

Declaration

```
// C#  
public byte[][] MessageId{get;}
```

Property Value

A byte[][] that specifies the message identifiers received as part of the notification.

Remarks

This property specifies the message identifiers of the messages that raise the notification.

The size of the MessageId array is 1 for regular notifications. The size of the MessageId array is 1 for grouping notifications if the notification grouping type is OracleAQNotificationGroupingType.Last. This property is not relevant if the NotificationType is OracleAQNotificationType.Timeout.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessageAvailableEventArgs Class](#)
- [OracleAQMessageAvailableEventArgs Members](#)

NotificationType

This property indicates the type of notification such as regular, grouping, or timeout.

Declaration

```
// C#  
public OracleAQNotificationType NotificationType {get;}
```

Property Value

An OracleAQNotificationType enum value.

Remarks

Regular, grouping, and timeout notifications are supported for AQ but only regular notification is supported for TxEventQ.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessageAvailableEventArgs Class](#)
- [OracleAQMessageAvailableEventArgs Members](#)

OriginalMessageId

This property specifies the ID of the message, in the last queue, that generated this message.

Declaration

```
// C#  
public byte[] OriginalMessageId {get;}
```

Property Value

A byte[] that specifies the original message ID.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessageAvailableEventArgs Class](#)
- [OracleAQMessageAvailableEventArgs Members](#)

Payload

This instance property provides payload associated with the notification.

Declaration

```
// C#  
public object Payload {get;}
```

Property Value

An object that represents payload associated with the notification.

Remarks

We receive payload only with TxEventQ notifications. In case of classic queue, we don't receive Payload as part of notification hence the Payload property will be Null for such scenario.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessageAvailableEventArgs Class](#)
- [OracleAQMessageAvailableEventArgs Members](#)

Priority

This instance property specifies the priority of the message.

Declaration

```
// C#  
public int Priority {get;}
```

Property Value

An integer that specifies the priority of the message.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessageAvailableEventArgs Class](#)
- [OracleAQMessageAvailableEventArgs Members](#)

Queue

This property indicates OracleAQQueue instance corresponding to which the notification is received.

Declaration

```
// C#  
public OracleAQQueue Queue { get;}
```

Property Value

An OracleAQQueue instance corresponding to which the notification is received

Remarks

This property is introduced from 23.4 release onwards. Till 23.3 release, OracleAQMessageAvailableEventArgs class had a property named QueueName, value of which was queue name corresponding to which the notification is received but this property is removed from 23.4 release onwards because OracleAQQueue class itself contains property Name which solves the purpose for former property.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessageAvailableEventArgs Class](#)
- [OracleAQMessageAvailableEventArgs Members](#)

QueueName

This property indicates the name of the queue that contains the message to be dequeued.

Declaration

```
// C#  
public string QueueName {get;}
```

Property Value

A string.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessageAvailableEventArgs Class](#)
- [OracleAQMessageAvailableEventArgs Members](#)

SenderId

This property identifies the original sender of the message.

Declaration

```
// C#  
public OracleAQAgent SenderId {get;}
```

Property Value

An OracleAQAgent object.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessageAvailableEventArgs Class](#)
- [OracleAQMessageAvailableEventArgs Members](#)

State

This instance property specifies the state of the message.

Declaration

```
// C#
public OracleAQMessageState State {get;}
```

Property Value

An OracleAQMessageState enumerated value.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQMessageAvailableEventArgs Class](#)
- [OracleAQMessageAvailableEventArgs Members](#)

OracleAQMessageAvailableEventHandler Delegate

The OracleAQMessageAvailableEventHandler delegate represents the signature of the method that handles the OracleAQQueue.MessageAvailable event.

Declaration

```
// C#
public delegate void OracleAQMessageAvailableEventHandler (object
    sender, OracleAQMessageAvailableEventArgs eventArgs);
```

Parameters

- *sender*
The source of the event.
- *eventArgs*
The OracleAQMessageAvailableEventArgs object that contains the event data.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- ["MessageAvailable Event"](#)

OracleAQQueue Class

An OracleAQQueue object represents a queue.

Class Inheritance

System.Object
 OracleAQQueue

Declaration

```
// C#  
public class OracleAQQueue : IDisposable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

A queue is a repository of messages and may either be a user queue, or an exception queue. A user queue is for normal message processing. A message is moved from a user queue to an exception queue if it cannot be retrieved and processed for some reason.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Members](#)
- [OracleAQQueue Constructors](#)
- [OracleAQQueue Static Methods](#)
- [OracleAQQueue Properties](#)
- [OracleAQQueue Public Methods](#)
- [OracleAQQueue Events](#)

OracleAQQueue Members

OracleAQQueue members are listed in the following tables.

OracleAQQueue Constructors

OracleAQQueue constructors are listed in [Table 12-20](#).

Table 12-20 OracleAQQueue Constructors

Constructor	Description
OracleAQQueue Constructors	Instantiate a new instance of the OracleAQQueue class (Overloaded).

OracleAQQueue Static Methods

The OracleAQQueue static method is listed in [Table 12-21](#).

Table 12-21 OracleAQQueue Static Methods

Static Method	Description
Listen	Listens for messages on one or more queues for one or more consumers, as specified in the array of OracleAQAgent objects (Overloaded).

OracleAQQueue Properties

OracleAQQueue properties are listed in [Table 12-22](#).

Table 12-22 OracleAQQueue Properties

Property	Description
AutoCommitTEQNotification	Used to automatically send a Commit acknowledgement after receiving a TxEventQ message's notification
Connection	Specifies the OracleConnection object associated with the queue.

Table 12-22 (Cont.) OracleAQQueue Properties

Property	Description
DequeueOptions	Specifies the dequeuing options to use when dequeuing a message from a queue.
EnqueueOptions	Specifies the enqueueing options used to enqueue a message to a queue.
MessageType	Specifies the type of queue table associated with this queue.
Name	Returns the name of the queue.
Notification	Specifies the various notification options for notifications that are registered using the <code>MessageAvailable</code> event.
NotificationConsumers	Specifies the array of consumers, for a multiconsumer queue, that are to be notified asynchronously for any incoming messages on the queue.
UdtTypeName	Specifies the type name on which the queue and the corresponding queue table is based if the <code>MessageType</code> is <code>OracleAQMessageType.UDT</code> .

OracleAQQueue Public Methods

The `OracleAQQueue` public methods are listed in [Table 12-23](#).

Table 12-23 OracleAQQueue Public Methods

Public Method	Description
AcknowledgeTxEventQNotification	Sends an acknowledgement for a <code>TxEvtQ</code> notification
Dequeue	Dequeues messages from queues (Overloaded).
DequeueArray	Dequeues multiple messages from queues (Overloaded).
Dispose	Releases any resources or memory allocated by the object
Enqueue	Enqueues messages to queues (Overloaded).
EnqueueArray	Enqueues multiple messages to a queue (Overloaded).
Listen	Listens for messages on the queue on behalf of <code>listenConsumers</code> (Overloaded).

OracleAQQueue Events

The `OracleAQQueue` event is listed in [Table 12-24](#).

Table 12-24 OracleAQQueue Events

Event Name	Description
MessageAvailable Event	Notifies when a message is available in the queue for <code>NotificationConsumers</code> .

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)

OracleAQQueue Properties

OracleAQQueue properties are listed in [Table 12-25](#).

Table 12-25 OracleAQQueue Properties

Property	Description
AutoCommitTEQNotification	Used to automatically send a Commit acknowledgement after receiving a TxEventQ message's notification
Connection	Specifies the OracleConnection object associated with the queue.
DequeueOptions	Specifies the dequeuing options to use when dequeuing a message from a queue.
EnqueueOptions	Specifies the enqueueing options used to enqueue a message to a queue.
MessageType	Specifies the type of queue table associated with this queue.
Name	Returns the name of the queue.
Notification	Specifies the various notification options for notifications that are registered using the MessageAvailable event.
NotificationConsumers	Specifies the array of consumers, for a multiconsumer queue, that are to be notified asynchronously for any incoming messages on the queue.
UdtTypeName	Specifies the type name on which the queue and the corresponding queue table is based if the MessageType is OracleAQMessageType.UDT.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

AutoCommitTEQNotification

This property is used to automatically send a Commit acknowledgement after receiving a TxEventQ message's notification.

Declaration

```
// C#  
public bool AutoCommitTEQNotification {get; set;}
```

Property Value

A Boolean value

Remarks

The default value is false.

If this instance property is set to true, then a Commit acknowledgement is sent to the server for every TxEventQ message notification received automatically. Once set to true, an OracleAQMessageAvailableEventArgs AcknowledgeTEQNotification call does not result in a change as all the messages will be in the Processed state now.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

Connection

This property specifies the OracleConnection object associated with the queue.

Declaration

```
// C#  
public OracleConnection Connection {get; set;}
```

Property Value

An OracleConnection object that indicates the connection associated with the queue.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

This connection must be opened before calling methods like Enqueue and Dequeue.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

DequeueOptions

This instance property specifies the dequeuing options to use when dequeuing a message from a queue.

Declaration

```
// C#  
public OracleAQDequeueOptions DequeueOptions {get; set}
```

Property Value

An OracleAQDequeueOptions object.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

The default value is an OracleAQDequeueOptions object with default property values. Setting this property to null resets all dequeue options to their default values.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

EnqueueOptions

This instance property specifies the enqueueing options used to enqueue a message to a queue.

Declaration

```
// C#  
public OracleAQEnqueueOptions EnqueueOptions {get; set}
```

Property Value

An OracleAQEnqueueOptions object.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

The default value is an OracleAQEnqueueOptions object with default property values. Setting this property to null resets all enqueue options to their default values.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

MessageType

This instance property specifies the type of queue table associated with this queue.

Declaration

```
// C#
public OracleAQMessageType MessageType {get; set;}
```

Property Value

An `OracleAQMessageType` enumerated value.

Exceptions

`ArgumentOutOfRangeException` - The type value specified is invalid.

`ObjectDisposedException` - The object is already disposed.

Remarks

The `MessageType` property also dictates the type of message payloads that are enqueued or dequeued from the queue. It is possible to enqueue a variety of payloads depending on the `MessageType`. All ODP.NET provider types support RAW, user-defined type, and XML data type payloads. Managed ODP.NET and ODP.NET Core also support JSON data type payloads. ODP.NET does not support other data types, such as Java Message Service data types.

[Table 12-26](#) lists the allowed payload types for various message types.

Table 12-26 Message Types and Payloads

OracleAQQueue.MessageType	Allowed OracleAQMessage.Payload type to Enqueue
<code>OracleAQMessageType.JSON</code>	Either <code>OracleString</code> or <code>string</code>
<code>OracleAQMessageType.Raw (default)</code>	<code>OracleBinary</code> , <code>byte[]</code>
<code>OracleAQMessageType.Xml</code>	<code>OracleXmlType</code> , <code>XmlReader</code> , <code>String</code> (well-formed XML, else exception is raised)
<code>OracleAQMessageType.UDT</code>	UDT Custom Object

[Table 12-27](#) lists the payload types for dequeued messages.

Table 12-27 Payload Types for Dequeued Messages

OracleAQQueue.MessageType	DequeueOptions.Provi derSpecificType	OracleAQMessage.Payload of dequeued message
<code>OracleAQMessageType.JSON</code>	true	<code>OracleString</code>

Table 12-27 (Cont.) Payload Types for Dequeued Messages

OracleAQQueue.MessageType	DequeueOptions.ProviderSpecificType	OracleAQMessage.Payload of dequeued message
OracleAQMessageType.JSON	false	string
OracleAQMessageType.Xml	true	OracleXmlType
OracleAQMessageType.Xml	false	XmlReader
OracleAQMessageType.Raw	true	OracleBinary
OracleAQMessageType.Raw	false	Byte[]
OracleAQMessageType.UDT	N.A.	UDT Custom Object

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

Name

This instance property returns the name of the queue.

Declaration

```
// C#
public string Name {get;}
```

Property Value

A string that indicates the name of the queue.

Exceptions

ObjectDisposedException - The object is already disposed.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

Notification

This instance property specifies the various notification options for notifications that are registered using the MessageAvailable event.

Declaration

```
// C#  
public OracleNotificationRequest Notification {get;}
```

Property Value

Specifies an `OracleNotificationRequest` object whose properties can be changed to alter the notification behavior.

Remarks

This property can be used to change various notification options. The notification options must be changed before registering with the `MessageAvailable` event. This property can be modified again only after unregistering from the `MessageAvailable` event.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

NotificationConsumers

This instance property specifies the array of consumers, for a multiconsumer queue, that are to be notified asynchronously for any incoming messages on the queue.

Declaration

```
// C#  
public string[] NotificationConsumers {get; set;}
```

Property Value

Specifies an array of consumer name strings for which the notifications are delivered.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - `MessageAvailable` registration is active.

Remarks

The consumer names must be in uppercase. This functionality only supports queues with uppercase names.

The list of consumers is used in the `MessageAvailable` event. The list must be set before registering for the event. This property cannot be modified after registering for the `MessageAvailable` event. This property can be modified again only after unregistering from `MessageAvailable` event.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

UdtTypeName

This instance property specifies the type name on which the queue and the corresponding queue table is based if the MessageType is `OracleAQMessageType.UDT`.

Declaration

```
// C#  
public string UdtTypeName {get; set;}
```

Property Value

Specifies the Oracle user-defined type name if the MessageType is `OracleAQMessageType.UDT`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The `UdtTypeName` property corresponds to the user-defined type name of the payload. This property must always be specified if the payload is a user-defined type. This property need not be set for other payload types.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

OracleAQQueue Constructors

`OracleAQQueue` constructors create new instances of the `OracleAQQueue` class.

Overload List:

- [OracleAQQueue\(string\)](#)
This constructor takes a queue name to initialize a queue object.
- [OracleAQQueue\(string, OracleConnection\)](#)
This constructor takes a queue name and connection to initialize a queue object. The connection does not need be open during the queue object construction.
- [OracleAQQueue\(string, OracleConnection, OracleAQMessageType\)](#)

This constructor takes a queue name, connection, and message type enumeration to initialize a queue object.

- [OracleAQQueue\(string, OracleConnection, OracleAQMessageType, string\)](#)

This constructor takes a queue name, connection, message type enumeration, and UDT type name to initialize a queue object.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

OracleAQQueue(string)

This constructor takes a queue name to initialize a queue object.

Declaration

```
// C#  
public OracleAQQueue(string name);
```

Parameters

- *name*
The name of the queue as specified in the database.

Exceptions

ArgumentNullException - The queue name is null.

ArgumentException - The queue name is empty.

Remarks

The operation of creating an *OracleAQQueue* object does not involve checking for the existence of the queue in the database.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

OracleAQQueue(string, OracleConnection)

This constructor takes a queue name and connection to initialize a queue object. The connection does not need to be open during the queue object construction.

Declaration

```
// C#  
public OracleAQQueue(string name, OracleConnection con);
```

Parameters

- *name*
Name of the queue as specified in the database.
- *con*
An OracleConnection object that connects to the queue.

Exceptions

ArgumentNullException - Either the connection is null or queue name is null.

ArgumentException - Queue name is empty.

Remarks

The connection can be accessed using the Connection property, and it must be opened before calling any operational APIs such as Enqueue and Dequeue.

Creating an OracleAQQueue object does not check for the existence of the queue in the database.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

OracleAQQueue(string, OracleConnection, OracleAQMessageType)

This constructor takes a queue name, connection and message type enumeration to initialize a queue object. The connection does not need to be open during the queue object construction.

Declaration

```
// C#  
public OracleAQQueue(string name, OracleConnection con, OracleAQMessageType  
    messageType);
```

Parameters

- *name*
The name of the queue as specified in the database.
- *con*
An OracleConnection object that is used to connect to the queue.
- *messageType*

An *OracleAQMessageType* enumeration specifying the type of the message that is enqueued or dequeued from this queue.

Exceptions

ArgumentNullException - Either the connection is null or queue name is null.

ArgumentException - Queue name is empty or the specified message type is not valid.

Remarks

Creating an *OracleAQQueue* object does not check for the existence of the queue in the database.

You need to set the *UdtTypeName* property before using the queue object if the *messageType* is a UDT. Another approach is to create a queue using the other constructor overload by supplying the *udtTypeName*.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

OracleAQQueue(string, OracleConnection, OracleAQMessageType, string)

This constructor takes a queue name, connection, message type enumeration, and UDT type name to initialize a queue object. The connection does not need to be open during the queue object construction.

Declaration

```
// C#
public OracleAQQueue(string name, OracleConnection con, OracleAQMessageType
    messageType, string udtTypeName);
```

Parameters

- *name*
The name of the queue as specified in the database.
- *con*
An *OracleConnection* object that is used to connect to the queue.
- *messageType*
An *OracleAQMessageType* enumeration specifying the type of the message that is enqueued or dequeued from this queue.
- *udtTypeName*
The name of the database object type used if the *messageType* is UDT. The *udtTypeName* parameter represents the type on which the queue is based.

Exceptions

ArgumentNullException - The connection is null or the queue name is null.
ArgumentException - The queue name is empty or the specified messageType is not valid.

Remarks

Creating an OracleAQQueue object does not check for the existence of the queue in the database.

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

OracleAQQueue Static Methods

OracleAQQueue static methods are listed in [Table 12-28](#).

Table 12-28 OracleAQQueue Static Methods

Static Method	Description
Listen	Listens for messages on one or more queues for one or more consumers, as specified in the array of OracleAQAgent objects (Overloaded).

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

Listen

Listen methods listen for messages on one or more queues for one or more consumers as specified in the array of OracleAQAgent objects.

Listen methods are supported only for AQ, but not for TxEventQ

Overload list

- [Listen\(OracleConnection, OracleAQAgent\[\]\)](#)
This static method listens for messages on one or more queues for one or more consumers as specified in the array of OracleAQAgent objects.

- [Listen\(OracleConnection, OracleAQAgent\[\], int\)](#)

This static method listens for messages on one or more queues for one or more consumers as specified in the array of `OracleAQAgent` objects. It also specifies a wait time.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

Listen(OracleConnection, OracleAQAgent[])

This static method listens for messages on one or more queues for one or more consumers as specified in the array of `OracleAQAgent` objects.

Declaration

```
// C#  
public static OracleAQAgent Listen(OracleConnection con, OracleAQAgent[]  
    listenConsumers);
```

Parameters

- *con*
An `OracleConnection` instance.
- *listenConsumers*
The array of consumers being listened for. The name of the `OracleAQAgent` object must be null or empty for single consumer queues.

Return Value

An `OracleAQAgent` object.

Exceptions

`ArgumentNullException` - The *con* or *listenConsumers* parameter is null.

`InvalidOperationException` - The connection is not open.

Remarks

`Listen` is useful in situations where one needs to monitor multiple queues until a message is available for a consumer in one of the queues. The `Name` property of the `OracleAQAgent` object represents the name of the consumer and the `Address` property represents the name of the queue.

This call blocks the calling thread until there is a message ready for consumption for a consumer in the list. It returns an `OracleAQAgent` object which specifies the consumer and queue for which a message is ready to be dequeued.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

Listen(OracleConnection, OracleAQAgent[], int)

This static method listens for messages on one or more queues for one or more consumers as specified in the array of `OracleAQAgent` objects. The `Name` property of the `OracleAQAgent` object represents the name of the consumer and the `Address` property of the `OracleAQAgent` object represents the name of the queue.

Declaration

```
// C#  
public static OracleAQAgent Listen(OracleConnection con, OracleAQAgent[]  
    listenConsumers, int waitTime);
```

Parameters

- *con*
An `OracleConnection` instance.
- *listenConsumers*
The array of consumers being listened for. The name of the `OracleAQAgent` object must be null or empty for single consumer queues.
- *waitTime*
Wait time in seconds.

Return Value

An `OracleAQAgent` object.

Exceptions

`ArgumentNullException` - The *con* or *listenConsumers* parameter is null.

`InvalidOperationException` - The connection is not open.

`ArgumentException` - *waitTime* is less than -1.

Remarks

In case of timeout, this method returns null.

`Listen` is useful in situations where one needs to monitor multiple queues until a message is available for a consumer in one of the queues. The `Name` property of the `OracleAQAgent` object represents the name of the consumer and the `Address` property of the `OracleAQAgent` object represents the name of the queue.

A *waitTime* of -1 implies an infinite wait time.

This call blocks the calling thread until there is a message ready for consumption for a consumer in the list. It returns an `OracleAQAgent` object which specifies the consumer and queue for which a message is ready to be dequeued.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

OracleAQQueue Public Methods

OracleAQQueue public methods are listed in [Table 12-29](#).

Table 12-29 OracleAQQueue Public Methods

Public Method	Description
AcknowledgeTxEventQNotification	Sends an acknowledgement for a TxEventQ notification
Dequeue	Dequeues messages from queues (Overloaded).
DequeueArray	Dequeues multiple messages from queues (Overloaded).
Dispose	Releases any resources or memory allocated by the object
Enqueue	Enqueues messages to queues (Overloaded).
EnqueueArray	Enqueues multiple messages to a queue (Overloaded).
Listen	Listens for messages on the queue on behalf of ListenConsumers (Overloaded).

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

AcknowledgeTxEventQNotification

This instance method sends an acknowledgement for a TxEventQ notification.

Declaration

```
// C#
public void AcknowledgeTxEventQNotification( OracleTxEventQNotificationDirective
directive);
```


Parameters

directive - Specifies the OracleTxEventQNotificationDirective enumeration value.

Exceptions

InvalidOperationException is thrown when:

- the CIC connection is closed,
- registration is not found, or
- acknowledgement is being sent to an AQ.

Remarks

This method should be called from a callback function after receiving a notification. There are two possible values for OracleTxEventQNotificationDirective enumeration, OracleTxEventQNotificationDirective.Commit or OracleTxEventQNotificationDirective.Rollback.

Acknowledgements are meant to be sent to TxEventQ only but not to an AQ queue. The callback method is invoked using a separate thread so its a good practice to catch exceptions otherwise the issues may go unnoticed.

In TxEventQ notifications, we get payload as part of the notification. In TxEventQ, each payload is associated with one of the three states below:

- Ready: This means message is available to be dequeued. As soon as a notification is sent for this message, the state gets changed to Acknowledgement_waiting.
- Acknowledgement_waiting: Notification for this message is sent and currently the TxEventQ is waiting for an acknowledgement from client application.
- Processed - This message is dequeued and it is not available for further dequeue.

When we send a Commit acknowledgement, all messages in the Acknowledgement_waiting state are marked as Processed.

When we send a Rollback acknowledgement, all messages in the Acknowledgement_waiting state are marked as Ready.

If message's state is Acknowledgement_waiting and the connection between application-server is terminated or crashed, then the state of these messages will automatically be reset as Ready.

Example

```

/*****Oracle DB Setup*****/

declare
    qprops          sys.dbms_aqadm.QUEUE_PROPS_T;
begin
    qprops.sort_list := 'enq_time';
    sys.dbms_aqadm.create_transactional_event_queue (queue_name =>
'raw_txeventq',
                                                    queue_payload_type => 'RAW',
                                                    queue_properties => qprops);

    END;
/

BEGIN

```

```

        dbms_aqadm.start_queue(
            queue_name => 'raw_txeventq');
    END;
    /
    *****/

// C#
using Oracle.ManagedDataAccess.Client;
using Oracle.ManagedDataAccess.Types;

namespace TxEventQ_notification_test
{
    class Program
    {
        bool IsNotified = false;
        static Random rnd = new Random();
        public static void Main(String[] args)
        {

            string constr = "user id=scott;password=tiger;data source=oracle";
            OracleConnection con = new OracleConnection(constr);
            {
                con.Open();

                using (OracleAQQueue queue = new OracleAQQueue("raw_txeventq", con,
OracleAQMessageType.Raw))
                {
                    try
                    {

                        queue.EnqueueOptions.Visibility = OracleAQVisibilityMode.Immediate;
                        queue.EnqueueOptions.DeliveryMode = OracleAQMessageDeliveryMode.Persistent;

                        Program obj = new Program();
                        queue.MessageAvailable += new
OracleAQMessageAvailableEventHandler(obj.OnMyNotificationReceived);

                        Console.WriteLine("Registration Done, please enqueue message inside
\\raw_txeventq\\ either from this application or any other application");

                        while(!obj.IsNotified)
                            Thread.Sleep(1000);

                        Console.WriteLine("Notification Received");

                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine(ex.ToString());
                    }
                }
            }

        }

        public void OnMyNotificationReceived(object src, OracleAQMessageAvailableEventArgs
arg)
        {
            Console.WriteLine("*****Notification Received*****");

            int tmp = rnd.Next(2);
            try

```

```

        {
            if (tmp == 0)
            {
                Console.WriteLine("Sending Commit Ack");

                arg.Queue.AcknowledgeTxEventQNotification(OracleTxEventQNotificationDirective.Commit);
            }
            else
            {
                Console.WriteLine("Sending Rollback Ack");

                arg.Queue.AcknowledgeTxEventQNotification(OracleTxEventQNotificationDirective.Rollback);
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine("????????????Exception in Callback Method????????????????\n" + ex.Message);
        }
        finally
        {
            {
                IsNotified = true;
            }
        }
    }
}
}

```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

Dequeue

Dequeue methods dequeue messages from queues.

Overload List

- [Dequeue\(\)](#)
This instance method dequeues messages from a queue using the DequeueOptions for the instance.
- [Dequeue\(OracleAQDequeueOptions\)](#)
This instance method dequeues messages from a queue using the supplied dequeue options.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

Dequeue()

This instance method is used to dequeue a message from a queue using the DequeueOptions for the instance.

Declaration

```
// C#  
public OracleAQMessage Dequeue();
```

Return Value

An OracleAQMessage instance representing the dequeued message.

Exceptions

InvalidOperationException - The connection is not open.

ObjectDisposedException - The object is already disposed.

OracleException - In case of timeout, an exception is thrown with the message, ORA-25228: timeout or end-of-fetch during message dequeue from queue_name. Timeout may happen if DequeueOptions.Wait is set to a value other than -1.

Remarks

The MessageType property must be set appropriately before calling this function. If the MessageType is OracleAQMessageType.UDT, then the UdtTypeName property must also be set.

Dequeued buffered messages always have null MessageId values.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

Dequeue(OracleAQDequeueOptions)

This instance method dequeues messages from a queue using the supplied dequeue options.

Declaration

```
// C#  
public OracleAQMessage Dequeue(OracleAQDequeueOptions dequeueOptions);
```

Parameters

- *dequeueOptions*
An OracleAQDequeueOptions object.

Return Value

An OracleAQMessage instance representing the dequeued message.

Exceptions

InvalidOperationException - The connection is not open.

ObjectDisposedException - The object is already disposed.

OracleException - In case of timeout, an exception is thrown with the message, ORA-25228: timeout or end-of-fetch during message dequeue from queue_name. Timeout may happen if DequeueOptions.Wait is set to a value other than -1.

Remarks

If the supplied dequeueOptions object is null, then the dequeue options default values are used. The queue object's DequeueOptions property is ignored for this operation.

Calling this method does not change the DequeueOptions property of the queue.

The MessageType property must be set appropriately before calling this function. If the MessageType is OracleAQMessageType.UDT, then the UdtTypeName property must also be set.

Dequeued buffered messages always have null MessageId values.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

DequeueArray

DequeueArray methods dequeue multiple messages from queues.

Overload List

- [DequeueArray\(int\)](#)
This instance method dequeues multiple messages from a queue using the DequeueOptions of the instance.
- [DequeueArray\(int, OracleAQDequeueOptions\)](#)
This instance method dequeues multiple messages from a queue using the supplied dequeue options.

Note

DequeueArray methods are not supported for JSON Data type for both AQ and TxEventQ, the workaround is to use non-array APIs with JSON type payloads.

DequeueMode = RemoveNoData is not supported for TxEventQ. For AQ, it is not supported only for JSON data type.

DequeueMode = Locked is not supported with Visibility = Immediate for AQ and TxEventQ, the workaround is to use DequeueMode = Locked with Visibility = OnCommit.

DeliveryMode = Buffered is not supported for AQ and TxEventQ operations, workaround is to use DeliveryMode = Persistent.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

DequeueArray(int)

This instance method dequeues multiple messages from a queue using the DequeueOptions of the instance.

Note

JSON payloads are not supported.

Declaration

```
// C#  
public OracleAQMessage[] DequeueArray(int dequeueCount);
```

Parameters

- *dequeueCount*
An integer specifying the numbers of messages to dequeue.

Return Value

An array of OracleAQMessage instances representing the dequeued messages.

Exceptions

ArgumentOutOfRangeException - dequeueCount is less than or equal to 0.

InvalidOperationException - The connection is not open.

ObjectDisposedException - The object is already disposed.

OracleException - In case of timeout, an exception is thrown with the message, ORA-25228: timeout or end-of-fetch during message dequeue from queue_name. Timeout may happen if DequeueOptions.Wait is set to a value other than -1.

Remarks

The MessageType property must be set appropriately before calling this function. If the MessageType is OracleAQMessageType.UDT, then the UdtTypeName property must be set as well.

The size of the returned array may be less than the dequeueCount. It depends on the actual number of messages present in the queue.

Dequeued buffered messages always have null MessageId values irrespective of the database version.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

DequeueArray(int, OracleAQDequeueOptions)

This instance method dequeues multiple messages from a queue using the supplied dequeue options.

① Note

JSON payloads are not supported.

Declaration

```
// C#  
public OracleAQMessage[] DequeueArray(int dequeueCount, OracleAQDequeueOptions  
dequeueOptions);
```

Parameters

- *dequeueCount*
An integer specifying the numbers of messages to dequeue.
- *dequeueOptions*
An OracleAQDequeueOptions object.

Return Value

An array of OracleAQMessage instances representing the dequeued messages.

Exceptions

`ArgumentOutOfRangeException` - `dequeueCount` is less than or equal to 0.

`InvalidOperationException` - The connection is not open.

`ObjectDisposedException` - The object is already disposed.

`OracleException` - In case of timeout, an exception is thrown with the message, ORA-25228: timeout or end-of-fetch during message dequeue from `queue_name`. Timeout may happen if `DequeueOptions.Wait` is set to a value other than -1.

Remarks

Calling this method does not change the `DequeueOptions` property of the queue.

If the supplied `dequeueOptions` object is null, then the dequeue options default values are used. The `DequeueOptions` property of the queue object is ignored in this operation.

The `MessageType` property must be set appropriately before calling this function. If the `MessageType` is `OracleAQMessageType.UDT`, then the `UdtTypeName` property must be set as well.

The size of the returned array may be less than the `dequeueCount`. It depends on the actual number of messages present in the queue.

Dequeued buffered messages always have null `MessageId` values irrespective of the database version.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

Dispose

This method releases any resources or memory allocated by the object.

Declaration

```
// C#  
public void Dispose();
```

Implements

`IDisposable`.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

Enqueue

Enqueue instance methods enqueue messages to queues.

Overload List

- [Enqueue\(OracleAQMessage\)](#)
This instance method enqueues messages to a queue using the EnqueueOptions of the instance.
- [Enqueue\(OracleAQMessage, OracleAQEnqueueOptions\)](#)
This instance method enqueues messages to a queue using the supplied enqueue options.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

Enqueue(OracleAQMessage)

This instance method enqueues messages to a queue using the EnqueueOptions of the instance.

Declaration

```
// C#  
public void Enqueue(OracleAQMessage message);
```

Parameters

- *message*
An OracleAQMessage object.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The connection is not open.

ArgumentNullException - The message parameter is null.

ArgumentException - The message payload is `OracleXmlType` and the connection used to create `OracleXmlType` is different from the queue's connection.

Remarks

`MessageId` of the enqueued message is populated after the call to `Enqueue` completes. Enqueued buffered messages always have null `MessageId` values.

The `MessageType` property needs to be set appropriately before calling this function. If the `MessageType` is `OracleAQMessageType.UDT`, then the `UdtTypeName` property must be set as well.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

Enqueue(OracleAQMessage, OracleAQEnqueueOptions)

This instance method enqueues messages to a queue using the supplied enqueue options.

Declaration

```
// C#  
public void Enqueue(OracleAQMessage message, OracleAQEnqueueOptions enqueueOptions);
```

Parameters

- *message*
An `OracleAQMessage` object.
- *enqueueOptions*
An `OracleAQEnqueueOptions` object.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The connection is not open.

ArgumentNullException - The message parameter is null.

ArgumentException - The message payload is `OracleXmlType` and the connection used to create `OracleXmlType` is different from the queue's connection.

Remarks

If the supplied `enqueueOptions` object is null, then the enqueue options default values are used. The `EnqueueOptions` property of the queue object is ignored in this operation.

The `MessageId` of the enqueued message is populated after the call to `Enqueue` completes. Enqueued buffered messages always have null `MessageId` values. Calling this method does not change the `EnqueueOptions` property of the queue.

The `MessageType` property must be set appropriately before calling this function. If the `MessageType` is `OracleAQMessageType.UDT`, then the `UdtTypeName` property must also be set.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

EnqueueArray

`EnqueueArray` instance methods enqueue multiple messages to a queue.

Overload List

- [EnqueueArray\(OracleAQMessage\[\]\)](#)
This instance method enqueues multiple messages to a queue using the `EnqueueOptions` of the instance.
- [EnqueueArray\(OracleAQMessage\[\], OracleAQEnqueueOptions\)](#)
This instance method enqueues multiple messages to a queue using the supplied enqueue options.

① Note

The enqueue array functionality is not supported for JSON data type for both AQ and TxEventQ, the workaround is to use non-array APIs for JSON payloads.

Enqueue Array with `Visibility = Immediate` is not supported for TxEventQ, the workaround is to use `Visibility = OnCommit`.

Buffered delivery mode is not supported for AQ and TxEventQ operations, the workaround is to use `DeliveryMode = Persistent`.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

EnqueueArray(OracleAQMessage[])

This instance method enqueues multiple messages to a queue using the `EnqueueOptions` of the instance.

Note

JSON payloads are not supported.

Declaration

```
// C#  
public int EnqueueArray(OracleAQMessage[] messages);
```

Parameters

- *messages*
An array of OracleAQMessage objects.

Return Value

An integer representing the number of messages actually enqueued.

Exceptions

ArgumentNullException - The message parameter is null.

ArgumentException - At least one of the OracleAQMessage[] elements is null, or at least one of the OracleAQMessage[] elements has a payload of OracleXmlType, which is created using a connection that is different from the queue's connection.

InvalidOperationException - The OracleAQMessage array is empty or the connection is not open.

ObjectDisposedException - The object is already disposed.

Remarks

The MessageId properties of the enqueued messages are populated after the call to Enqueue completes. Enqueued buffered messages always have null MessageId values.

The MessageType property must be set appropriately before calling this function. If the MessageType is OracleAQMessageType.UDT, then the UdtTypeName property must also be set.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

EnqueueArray(OracleAQMessage[], OracleAQEnqueueOptions)

This instance method enqueues multiple messages to a queue using the supplied enqueue options.

Note

JSON payloads are not supported.

Declaration

```
// C#  
public int EnqueueArray(OracleAQMessage[] messages, OracleAQEnqueueOptions  
    enqueueOptions);
```

Parameters

- *messages*
An array of OracleAQMessage objects.
- *enqueueOptions*
An OracleAQEnqueueOptions object.

Return Value

An integer representing the number of messages actually enqueued.

Exceptions

ArgumentNullException - The message parameter is null.

ArgumentException - At least one of the OracleAQMessage[] elements is null, or at least one of the OracleAQMessage[] elements has a payload of OracleXmlType, which is created using a connection that is different from the queue's connection.

InvalidOperationException - The OracleAQMessage array is empty or the connection is not open.

ObjectDisposedException - The object is already disposed.

Remarks

MessageId properties of the enqueued messages are populated after the call to Enqueue completes. Enqueued buffered messages always have null MessageId values. Calling this method does not change the EnqueueOptions property of the queue.

If the supplied enqueueOptions object is null, then the enqueue options default values are used. The EnqueueOptions property of the queue object is ignored in this operation.

The MessageType property must be set appropriately before calling this function. If the MessageType is OracleAQMessageType.UDT, then the UdtTypeName property must also be set.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

Listen

Listen methods listen for messages on the queue on behalf of `listenConsumers`.

Listen methods are supported for AQ, but not for `TxEvtQ`.

Overload List

- [Listen\(string\[\]\)](#)
This method listens for messages on the queue on behalf of `listenConsumers`.
- [Listen \(string\[\], int\)](#)
This method listens for messages on behalf of `listenConsumers` for a specified time.

📘 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

Listen(string[])

This method listens for messages on the queue on behalf of `listenConsumers`.

Declaration

```
// C#  
public string Listen(string[] listenConsumers);
```

Parameters

- *listenConsumers*
An array of consumers to listen for on this queue. This parameter should be null in case of single consumer queues.

Return Value

A string.

Exceptions

`InvalidOperationException` - The connection is not open.

`ObjectDisposedException` - The object is already disposed.

Remarks

This call blocks the calling thread until there is a message ready for consumption for a consumer in the `listenConsumers` array. It returns a string representing the consumer name for which the message is ready.

`Listen` is useful in situations that require waiting until a message is available in the queue for consumers whose names are specified in `listenConsumers`.

Example

The following example demonstrates using the Listen method. The first part of the example performs the requisite database setup for the database user, SCOTT. The second part of the example demonstrates how a thread can listen and wait until a message is enqueued.

```
-- Part I: Database setup required for this demo

-----
-- SQL to grant appropriate privilege to database user, SCOTT
-----
SQL> ALTER USER SCOTT ACCOUNT UNLOCK IDENTIFIED BY Pwd4Sct;
User altered.
GRANT ALL ON DBMS_AQADM TO scott;

-----
-- PLSQL to create queue-table and queue and start queue for SCOTT
-----
BEGIN
  DBMS_AQADM.CREATE_QUEUE_TABLE(
    queue_table=>'scott.test_q_tab',
    queue_payload_type=>'RAW',
    multiple_consumers=>FALSE);

  DBMS_AQADM.CREATE_QUEUE(
    queue_name=>'scott.test_q',
    queue_table=>'scott.test_q_tab');

  DBMS_AQADM.START_QUEUE(queue_name=>'scott.test_q');
END;
/

-----
-- PLSQL to stop queue and drop queue & queue-table from SCOTT
-----
BEGIN
  DBMS_AQADM.STOP_QUEUE('scott.test_q');

  DBMS_AQADM.DROP_QUEUE(
    queue_name => 'scott.test_q',
    auto_commit => TRUE);

  DBMS_AQADM.DROP_QUEUE_TABLE(
    queue_table => 'scott.test_q_tab',
    force => FALSE,
    auto_commit => TRUE);
END;
/
-- End of Part I, database setup.

//Part II: Demonstrates using the Listen method
//C#
using System;
using System.Text;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;
using System.Threading;

namespace ODPSample
{
  /// <summary>
```

```

/// Demonstrates how a thread can listen and wait until a message is enqueued.
/// Once a message is enqueued, the listening thread returns from the
/// blocked Listen() method invocation and dequeues the message.
/// </summary>
class EnqueueDequeue
{
    static bool s_bListenReturned = false;

    static void Main(string[] args)
    {
        // Create connection
        string constr = "user id=scott;password=Pwd4Sct;data source=oracle";
        OracleConnection con = new OracleConnection(constr);

        // Create queue
        OracleAQQueue queue = new OracleAQQueue("scott.test_q", con);

        try
        {
            // Open connection
            con.Open();

            // Set message type for the queue
            queue.MessageType = OracleAQMessageType.Raw;

            // Spawning a thread which will listen for a message
            ThreadStart ts = new ThreadStart(TestListen);
            Thread t = new Thread(ts);
            t.Start();

            System.Threading.Thread.Sleep(2000);

            // Begin transaction for enqueue
            OracleTransaction txn = con.BeginTransaction();

            // Prepare message and RAW payload
            OracleAQMessage enqMsg = new OracleAQMessage();
            byte[] bytePayload = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
            enqMsg.Payload = bytePayload;

            // Prepare to Enqueue
            queue.EnqueueOptions.Visibility = OracleAQVisibilityMode.OnCommit;

            Console.WriteLine("[Main Thread]   Enqueuing a message...");
            Console.WriteLine("[Main Thread]   Enqueued Message Payload      : "
                + ByteArrayToString(enqMsg.Payload as byte[]));
            Console.WriteLine();

            // Enqueue message
            queue.Enqueue(enqMsg);

            // Enqueue transaction commit
            txn.Commit();

            // Loop till Listen returns
            while (!s_bListenReturned)
                System.Threading.Thread.Sleep(1000);
        }
        catch (Exception e)
        {
            Console.WriteLine("Error: {0}", e.Message);
        }
    }
}

```



```

        finally
        {
            // Close/Dispose objects
            queue.Dispose();
            con.Close();
            con.Dispose();
        }
    }

    static void TestListen()
    {
        // Create connection
        string constr = "user id=scott;password=Pwd4Sct;data source=oracle";
        OracleConnection conListen = new OracleConnection(constr);

        // Create queue
        OracleAQQueue queueListen = new OracleAQQueue("scott.test_q", conListen);

        try
        {
            // Open the connection for Listen thread.
            // Connection blocked on Listen thread can not be used for other DB
            // operations
            conListen.Open();

            // Set message type for the queue
            queueListen.MessageType = OracleAQMessageType.Raw;

            // Listen
            queueListen.Listen(null);

            Console.WriteLine("[Listen Thread] Listen returned... Dequeueing...");

            // Begin txn for Dequeue
            OracleTransaction txn = conListen.BeginTransaction();

            // Prepare to Dequeue
            queueListen.DequeueOptions.Visibility = OracleAQVisibilityMode.OnCommit;
            queueListen.DequeueOptions.Wait = 10;

            // Dequeue message
            OracleAQMessage deqMsg = queueListen.Dequeue();
            Console.WriteLine("[Listen Thread] Dequeued Message Payload      : "
                + ByteArrayToString(deqMsg.Payload as byte[]));

            // Dequeue txn commit
            txn.Commit();

            // Allow the main thread to exit
            s_bListenReturned = true;
        }
        catch (Exception e)
        {
            Console.WriteLine("Error: {0}", e.Message);
        }
        finally
        {
            // Close/Dispose objects
            queueListen.Dispose();
            conListen.Close();
            conListen.Dispose();
        }
    }

```

```
    }

    // Function to convert byte[] to string
    static private string ByteArrayToString(byte[] byteArray)
    {
        StringBuilder sb = new StringBuilder();
        for (int n = 0; n < byteArray.Length; n++)
        {
            sb.Append((int.Parse(byteArray[n].ToString())).ToString("X"));
        }
        return sb.ToString();
    }
}
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

Listen (string[], int)

This method listens for messages on behalf of `listenConsumers` for a specified time.

Declaration

```
// C#
public string Listen(string[] listenConsumers, int waitTime);
```

Parameters

- *listenConsumers*
Array of consumers for which to listen on this queue.
- *waitTime*
Wait time in seconds.

Return Value

A string

Exceptions

`InvalidOperationException` - The connection is not open.

`ArgumentException` - `waitTime` is less than -1.

`ObjectDisposedException` - The object is already disposed.

Remarks

`Listen` is useful in situations that require waiting until a message is available in the queue for consumers whose names are specified in `listenConsumers`.

This call blocks the calling thread until there is a message ready for consumption for a consumer in the `listenConsumers` array. It returns a string representing the consumer name for which the message is ready. The method returns null if a timeout occurs.

The `listenConsumers` parameter should be null for single consumer queues. An empty string is returned in such cases.

A `waitTime` of -1 implies infinite wait time.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

OracleAQQueue Events

The `OracleAQQueue` event is listed in [Table 12-30](#).

Table 12-30 OracleAQQueue Events

Event Name	Description
MessageAvailable Event	Notifies when a message is available in the queue for <code>NotificationConsumers</code> .

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)

MessageAvailable Event

This event is notified when a message is available in the queue for `NotificationConsumers`.

`TxEvtQ` notifications are not supported with ODP.NET Unmanaged Driver.

Declaration

```
// C#  
public event OracleAQMessageAvailableEventHandler MessageAvailable;
```

Event Data

The event handler receives an `OracleAQMessageAvailableEventArgs` object.

Exceptions

InvalidOperationException - The connection is not open.

Remarks

Asynchronous notification is supported in all queue tables.

In order to receive the notification about message availability, the client must create an `OracleAQMessageAvailableEventHandler` delegate to listen to this event. The delegate should be added to this event only after setting the [NotificationConsumers](#) and [Notification](#) properties.

The notification registration takes place after the first delegate is added to the event. The notification is unregistered when the last delegate is removed from the event. Notifications set on an `OracleAQQueue` object get cancelled automatically when the object gets disposed.

When registering for notifications with classic queue, commonly referred to as AQ, Oracle Data Provider for .NET opens a port to listen for notifications. HA events, load balancing, and continuous query notification features also share the same port. This port can be configured centrally by setting the database notification port in an application or Web configuration file. The following example code specifies a port number of 1200:

```
<configuration>
  <oracle.dataaccess.client>
    <settings>
      <add name="DbNotificationPort" value="1200"/>
    </settings>
  </oracle.dataaccess.client>
</configuration>
```

If the configuration file does not exist or the db notification port is not specified, then ODP.NET uses a valid and random port number. The configuration file may also request for a random port number by specifying a db notification port value of -1.

The notification listener, which runs in the same application domain as ODP.NET, uses the specified port number to listen to notifications from the database. A notification listener gets created when the application registers with `OracleAQQueue.MessageAvailable` event. One notification listener can listen to all notification types. Only one notification listener is created for each application domain.

When connecting to Transactional Event Queue (TxEventQ), ODP.NET will not use a listener nor open a port to receive notification. Instead, it will use a Client Initiated Connection(CIC). CIC uses a dedicated connection instead of the listening end point. It does not count toward Min Pool Size and Max Pool Size limits.

When notifications are registered with Classic Queues(AQ), all registrations get notification when a message is available but in case of TxEventQ, only one of the registrations get notification because it contains Payload also which can be dequeued from the notification itself upon sending a "Commit" Acknowledgement.

Example

The following example demonstrates application notification. The first part of the example performs the requisite database setup for the database user, SCOTT. The second part of the example demonstrates how an application is notified when a message is available in the queue.

-- Part I: Database setup required for this demo

```

-- SQL to grant appropriate privilege to database user, SCOTT
-----
SQL> ALTER USER SCOTT ACCOUNT UNLOCK IDENTIFIED BY Pwd4Sct;
User altered.
SQL> GRANT ALL ON DBMS_AQADM TO scott;

-----
-- PLSQL to create queue-table and queue and start queue for SCOTT
-----
BEGIN
  DBMS_AQADM.CREATE_QUEUE_TABLE(
    queue_table=>'scott.test_q_tab',
    queue_payload_type=>'RAW',
    multiple_consumers=>FALSE);

  DBMS_AQADM.CREATE_QUEUE(
    queue_name=>'scott.test_q',
    queue_table=>'scott.test_q_tab');

  DBMS_AQADM.START_QUEUE(queue_name=>'scott.test_q');
END;
/

-----
-- PLSQL to stop queue and drop queue & queue-table from SCOTT
-----
BEGIN
  DBMS_AQADM.STOP_QUEUE('scott.test_q');

  DBMS_AQADM.DROP_QUEUE(
    queue_name => 'scott.test_q',
    auto_commit => TRUE);

  DBMS_AQADM.DROP_QUEUE_TABLE(
    queue_table => 'scott.test_q_tab',
    force => FALSE,
    auto_commit => TRUE);
END;
/
-- End of Part I, database setup.

//Part II: Demonstrates application notification
//C#
using System;
using System.Text;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

namespace ODPSample
{
  /// <summary>
  /// Demonstrates how the application can be notified when a message is
  /// available in a queue.
  /// </summary>
  class Notification
  {
    static bool isNotified = false;

    static void Main(string[] args)
    {
      // Create connection
      string constr = "user id=scott;password=Pwd4Sct;data source=oracle";

```

```
OracleConnection con = new OracleConnection(constr);

// Create queue
OracleAQQueue queue = new OracleAQQueue("scott.test_q", con);

try
{
    // Open connection
    con.Open();

    // Set message type for the queue
    queue.MessageType = OracleAQMessageType.Raw;

    // Add the event handler to handle the notification. The
    // MsgReceived method will be invoked when a message is enqueued
    queue.MessageAvailable +=
        new OracleAQMessageAvailableEventHandler(Notification.MsgReceived);

    Console.WriteLine("Notification registered...");

    // Begin txn for enqueue
    OracleTransaction txn = con.BeginTransaction();

    Console.WriteLine("Now enqueueing message...");

    // Prepare message and RAW payload
    OracleAQMessage enqMsg = new OracleAQMessage();
    byte[] bytePayload = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
    enqMsg.Payload = bytePayload;

    // Prepare to Enqueue
    queue.EnqueueOptions.Visibility = OracleAQVisibilityMode.OnCommit;

    // Enqueue message
    queue.Enqueue(enqMsg);

    Console.WriteLine("Enqueued Message Payload      : "
        + ByteArrayToString(enqMsg.Payload as byte[]));
    Console.WriteLine("MessageId of Enqueued Message : "
        + ByteArrayToString(enqMsg.MessageId));
    Console.WriteLine();

    // Enqueue txn commit
    txn.Commit();

    // Loop while waiting for notification
    while (isNotified == false)
    {
        System.Threading.Thread.Sleep(2000);
    }
}
catch (Exception e)
{
    Console.WriteLine("Error: {0}", e.Message);
}
finally
{
    // Close/Dispose objects
    queue.Dispose();
    con.Close();
    con.Dispose();
}
```

```

    }

    static void MsgReceived(object src, OracleAQMessageAvailableEventArgs arg)
    {
        try
        {
            Console.WriteLine("Notification Received...");
            Console.WriteLine("QueueName : {0}", arg.QueueName);
            Console.WriteLine("Notification Type : {0}", arg.NotificationType);

            //following type-cast to "byte[]" is required only for .NET 1.x
            byte[] notifiedMsgId = (byte[]) arg.MessageId[0];
            Console.WriteLine("MessageId of Notified Message : "
                + ByteArrayToString(notifiedMsgId));
            isNotified = true;
        }
        catch (Exception e)
        {
            Console.WriteLine("Error: {0}", e.Message);
        }
    }

    // Function to convert byte[] to string
    static private string ByteArrayToString(byte[] byteArray)
    {
        StringBuilder sb = new StringBuilder();
        for (int n = 0; n < byteArray.Length; n++)
        {
            sb.Append((int.Parse(byteArray[n].ToString())).ToString("X"));
        }
        return sb.ToString();
    }
}
}
}

```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleAQQueue Class](#)
- [OracleAQQueue Members](#)
- [Continuous Query Notification Support](#)
- [AcknowledgeTxEventQNotification](#) method, for more information about Acknowledgement

OracleAQDequeueMode Enumeration

[Table 12-31](#) lists all the OracleAQDequeueMode enumeration values with a description of each enumerated value.

Table 12-31 OracleAQDequeueMode Members

Member Name	Description
Browse	Reads the message without acquiring any lock on the message. This is equivalent to a SELECT statement.
Locked	Reads and obtains a write lock on the message. The lock lasts for the duration of the transaction. This is equivalent to a SELECT FOR UPDATE statement.
Remove	Reads the message and updates or deletes it. This is the default. The message can be retained in the queue table based on the retention properties
RemoveNoData	Confirms receipt of the message but does not deliver the actual message content.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- "[OracleAQDequeueOptions Class](#)"
- "[DequeueMode](#)"

OracleAQMessageDeliveryMode Enumeration

The OracleAQMessageDeliveryMode enumeration type specifies the delivery mode of the message.

[Table 12-32](#) lists all the OracleAQMessageDeliveryMode enumeration values with a description of each enumerated value.

Table 12-32 OracleAQMessageDeliveryMode Members

Member Name	Description
Buffered	Indicates a buffered message. Both enqueue and dequeue buffered messaging operations must be in IMMEDIATE visibility mode. This means that these operations cannot be part of another transaction. You cannot specify delay when enqueueing buffered messages. Dequeueing applications can choose to dequeue persistent messages only, buffered messages only, or both types. Buffered messages can be queried using the AQ\$Queue_Table_Name view. These messages appear with states, IN-MEMORY or SPILLED. Transaction grouping queues and array enqueues are not supported for buffered messages in Oracle Database 11g release 1 (11.1) . One can still use the array enqueue procedure to enqueue buffered messages, but the array size must be set to 1. Array dequeue is not supported for buffered messaging, but one can still use the array dequeue procedure by setting array size to 1. Buffered messaging is faster than persistent messaging. Use buffered messaging for applications that do not require the reliability and transaction support of Oracle Database AQ persistent messaging.
Persistent	Indicates a persistent message. Persistent messaging ensures reliability and support transactions. It is slower than buffered messaging.
PersistentOrBuffered	Indicates a persistent or buffered message. This is used with Dequeue() when a consumer wants to dequeue a message irrespective of whether it is Persistent or Buffered.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- "[OracleAQDequeueOptions Class](#)"
- "[DeliveryMode](#)"

OracleAQMessageState Enumeration

The OracleAQMessageState enumeration type identifies the state of the message at the time of dequeue.

[Table 12-33](#) lists all the OracleAQMessageState enumeration values with a description of each enumerated value.

Table 12-33 OracleAQMessageState Members

Member Name	Description
Expired	Indicates that the message has been moved to the exception queue.
Processed	Indicates that the message has been processed and retained.
Ready	Indicates that the message is ready to be processed.
Waiting	Indicates that the message delay has not been reached.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- ["OracleAQMessage Class"](#)
- ["State"](#)

OracleAQMessageType Enumeration

The OracleAQMessageType enumeration type specifies the message payload type.

[Table 12-34](#) lists all the OracleAQMessageType enumeration values with a description of each enumerated value.

Table 12-34 OracleAQMessageType Members

Member Name	Description
Json	Indicates the Json message type. The data type of the payload must be either <code>OracleString</code> or <code>string</code> .
Raw	Indicates the Raw message type (default). The data type of the payload must be either <code>OracleBinary</code> or <code>byte[]</code> to enqueue the message.
Udt	Indicates the Oracle UDT message type. The ODP.NET AQ implementation currently does not support user defined types with LOB attributes. It also does not support other variants of user defined types such as VARRAY and nested tables, as Oracle Database AQ does not support them inherently.
Xml	Indicates the XML message type. The data type of the payload must be <code>OracleXmlType</code> , <code>XmlReader</code> , or <code>String</code> in order to enqueue the message. If the data type is <code>String</code> , it must be well-formed XML, else an exception is raised when enqueueing the message.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	<code>Oracle.DataAccess.dll</code>	<code>Oracle.ManagedDataAccess.dll</code>	<code>Oracle.ManagedDataAccess.dll</code>
Namespace	<code>Oracle.DataAccess.Client</code>	<code>Oracle.ManagedDataAccess.Client</code>	<code>Oracle.ManagedDataAccess.Client</code>
.NET Framework	4.8	4.8	-
.NET (Core)	-	-	See System Requirements

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- "[OracleAQQueue Class](#)"
- "[OracleAQQueue Constructors](#)"
- "[MessageType](#)"

OracleAQNavigationMode Enumeration

[Table 12-35](#) lists all the `OracleAQNavigationMode` enumeration values with a description of each enumerated value.

Table 12-35 OracleAQNavigationMode Members

Member Name	Description
FirstMessage	Retrieves the first message that is available and matches the search criteria. This resets the position to the beginning of the queue.
FirstMessageMultiGroup	Indicates that a call to DequeueArray resets the position to the beginning of the queue, and dequeues messages that are available and match the search criteria. Messages are dequeued till the dequeueCount limit is reached. The dequeued messages can belong to different transaction groups. You can use the OracleAQMessage.TransactionGroup property to distinguish between messages from different transaction groups. All messages from the same transaction group have the same value for the OracleAQMessage.TransactionGroup property.
NextMessage	Retrieves the next message that is available and matches the search criteria. If the previous message belongs to a message group, AQ retrieves the next available message that matches the search criteria and belongs to the message group.
NextMessageMultiGroup	Indicates that a call to DequeueArray dequeues the next set of messages that are available and match the search criteria. Messages are dequeued till the dequeueCount limit is reached. The dequeued messages can belong to different transaction groups. You can use the OracleAQMessage.TransactionGroup property to distinguish between messages from different transaction groups. All messages from the same transaction group have the same value for the OracleAQMessage.TransactionGroup property.
NextTransaction	Skips the remainder of the current transaction group (if any) and retrieves the first message of the next transaction group.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- "[OracleAQDequeueOptions Class](#)"
- "[NavigationMode](#)"

OracleAQNotificationGroupingType Enumeration

The OracleAQNotificationGroupingType enumeration type specifies the notification grouping type.

[Table 12-36](#) lists all the OracleAQNotificationGroupingType enumeration values with a description of each enumerated value.

Table 12-36 OracleAQNotificationGroupingType Members

Member Name	Description
Last	Indicates that only details of the last message in the notification group are provided.
Summary	Indicates that the Summary of all messages in the notification group is provided.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- "[OracleNotificationRequest Class](#)"

OracleAQNotificationType Enumeration

The OracleAQNotificationType enumeration type specifies the notification type of the received notification.

[Table 12-37](#) lists all the OracleAQNotificationType enumeration values with a description of each enumerated value.

Table 12-37 OracleAQNotificationType Members

Member Name	Description
Group	Indicates that the received notification is a grouping notification.

Table 12-37 (Cont.) OracleAQNotificationType Members

Member Name	Description
Regular	Indicates that the received notification is a regular notification.
Timeout	Indicates that the received notification is raised due to a timeout.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"

OracleAQVisibilityMode Enumeration

[Table 12-38](#) lists all the OracleAQVisibilityMode enumeration values with a description of each enumerated value.

Table 12-38 OracleAQVisibilityMode Members

Member Name	Description
Immediate	Indicates that the enqueue or dequeue operation is not part of the current transaction. The operation constitutes a transaction of its own.
OnCommit	Indicates that the enqueue or dequeue operation is part of the current transaction. This is the default case.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"

OracleTxEventQNotificationDirective Enumeration

The OracleTxEventQNotificationDirective enumeration type specifies the TxEventQ notification's acknowledgement directive value.

[Table 12-39](#) lists all the OracleTxEventQNotificationDirective enumeration values with a description of each enumerated value.

Table 12-39 OracleTxEventQNotificationDirective Members

Member Name	Description
Commit	A Commit acknowledgement indicates this message and all prior enqueued messages that are associated with the Acknowledgement_waiting state will have their state value set to Processed.
Rollback	A Rollback acknowledgement indicates this message and all prior enqueued messages that are associated with the Acknowledgement_waiting state will have their state value reset to Ready.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"

Oracle Data Provider for .NET Types Classes

This chapter describes the large object and REF CURSOR objects provided by Oracle Data Provider for .NET.

This chapter contains these topics:

- ODP.NET Types (ODP.NET LOB objects) consisting of these object classes:
 - [OracleBFile Class](#)
 - [OracleBlob Class](#)
 - [OracleClob Class](#)
- [OracleRefCursor Class](#)

All offsets are 0-based for all ODP.NET LOB object parameters.

OracleBFile Class

An OracleBFile is an object that has a reference to BFILE data. It provides methods for performing operations on BFILES.

 **Note**

OracleBFile is supported for applications running against Oracle8.x and later.

Class Inheritance

System.Object
System.MarshalByRefObject
System.IO.Stream
Oracle.DataAccess.Types.OracleBFile

Declaration

```
// C#  
public sealed class OracleBFile : Stream, ICloneable, INullable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

OracleBFile is supported for applications running against Oracle8.x and later.

Example

```
// Database Setup, if you have not done so yet.
/* Log on as DBA (SYS or SYSTEM) that has CREATE ANY DIRECTORY privilege.

CREATE OR REPLACE DIRECTORY MYDIR AS 'C:\TEMP';

*/

// C#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class OracleBFileSample
{
    static void Main()
    {
        // Create MYDIR directory object as indicated previously and create a file
        // MyFile.txt with the text ABCDABC under C:\TEMP directory.
        // Note that the byte representation of the ABCDABC is 65666768656667

        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleBFile bFile = new OracleBFile(con, "MYDIR", "MyFile.txt");

        // Open the OracleBFile
        bFile.OpenFile();

        // Read 7 bytes into readBuffer, starting at buffer offset 0
        byte[] readBuffer = new byte[7];
        int bytesRead = bFile.Read(readBuffer, 0, 7);

        // Prints "bytesRead = 7"
        Console.WriteLine("bytesRead = " + bytesRead);

        // Prints "readBuffer = 65666768656667"
        Console.Write("readBuffer = ");
        for(int index = 0; index < readBuffer.Length; index++)
        {
```

```
        Console.Write(readBuffer[index]);
    }
    Console.WriteLine();

    // Search for the 2nd occurrence of a byte pattern {66,67}
    // starting from byte offset 1 in the OracleBFile
    byte[] pattern = new byte[2] {66, 67};
    long posFound = bFile.Search(pattern, 1, 2);

    // Prints "posFound    = 6"
    Console.WriteLine("posFound    = " + posFound);

    // Close the OracleBFile
    bFile.CloseFile();

    bFile.Close();
    bFile.Dispose();

    con.Close();
    con.Dispose();
}
}
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Members](#)
- [OracleBFile Constructors](#)
- [OracleBFile Static Fields](#)
- [OracleBFile Static Methods](#)
- [OracleBFile Instance Properties](#)
- [OracleBFile Instance Methods](#)

OracleBFile Members

OracleBFile members are listed in the following tables.

OracleBFile Constructors

OracleBFile constructors are listed in [Table 13-1](#).

Table 13-1 OracleBFile Constructors

Constructor	Description
OracleBFile Constructors	Creates an instance of the OracleBFile class (Overloaded)

OracleBFile Static Fields

OracleBFile static fields are listed in [Table 13-2](#).

Table 13-2 OracleBFile Static Fields

Field	Description
MaxSize	The static field holds the maximum number of bytes a BFILE can hold, which is 4,294,967,295 ($2^{32} - 1$) bytes
Null	Represents a null value that can be assigned to the value of an OracleBFile instance

OracleBFile Static Methods

OracleBFile static methods are listed in [Table 13-3](#).

Table 13-3 OracleBFile Static Methods

Methods	Description
Equals	Inherited from System.Object (Overloaded)

OracleBFile Instance Properties

OracleBFile instance properties are listed in [Table 13-4](#).

Table 13-4 OracleBFile Instance Properties

Properties	Description
CanRead	Indicates whether or not the LOB stream can be read
CanSeek	Indicates whether or not forward and backward seek operations can be performed
CanWrite	Indicates whether or not the LOB object supports writing
Connection	Indicates the connection used to read from a BFILE
DirectoryName	Indicates the directory alias of the BFILE
FileExists	Indicates whether or not the specified BFILE exists
FileName	Indicates the name of the BFILE
IsEmpty	Indicates whether the BFILE is empty or not
IsNull	Indicates whether or not the current instance has a null value
IsOpen	Indicates whether the BFILE has been opened by this instance or not
Length	Indicates the size of the BFILE data in bytes
Position	Indicates the current read position in the LOB stream
Value	Returns the data, starting from the first byte in BFILE, as a byte array

OracleBFile Instance Methods

OracleBFile instance methods are listed in [Table 13-5](#).

Table 13-5 OracleBFile Instance Methods

Methods	Description
BeginRead	Inherited from <code>System.IO.Stream</code>
BeginWrite	<i>Not Supported</i>
Clone	Creates a copy of an <code>OracleBFile</code> object
Close	Closes the current stream and releases any resources associated with the stream
CloseFile	Closes the BFILE referenced by the current BFILE instance
Compare	Compares data referenced by the two <code>OracleBFiles</code>
CreateObjRef	Inherited from <code>System.MarshalByRefObject</code>
CopyTo	Copies data as specified (Overloaded)
CopyToAsync	Returns a Task-based asynchronous version of <code>OracleBFile.CopyTo()</code> , which copies data from the current instance to the provided object. (Overloaded)
Dispose	Releases resources allocated by this object
EndRead	Inherited from <code>System.IO.Stream</code>
EndWrite	<i>Not Supported</i>
Equals	Inherited from <code>System.Object</code> (Overloaded)
Flush	<i>Not Supported</i>
FlushAsync	<i>Not Supported</i>
GetHashCode	Inherited from <code>System.Object</code>
GetLifetimeService	Inherited from <code>System.MarshalByRefObject</code>
GetType	Inherited from <code>System.Object</code>
InitializeLifetimeService	Inherited from <code>System.MarshalByRefObject</code>
IsEqual	Compares the LOB references
OpenFile	Opens the BFILE specified by the <code>FileName</code> and <code>DirectoryName</code>
Read	Reads a specified amount of bytes from the <code>OracleBFile</code> instance and populates the buffer
ReadAsync	Returns a Task-based asynchronous version of <code>OracleBFile.Read()</code> , which reads a specified number of bytes from the <code>OracleBFile</code> instance and populates the buffer (Overloaded)
ReadByte	Inherited from <code>System.IO.Stream</code>
Search	Searches for a binary pattern in the current instance of an <code>OracleBFile</code>
Seek	Sets the position on the current LOB stream
SetLength	<i>Not Supported</i>
ToString	Inherited from <code>System.Object</code>
Write	<i>Not Supported</i>
WriteByte	<i>Not Supported</i>

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Members](#)

OracleBFile Constructors

OracleBFile constructors create new instances of the OracleBFile class.

Overload List:

- [OracleBFile\(OracleConnection\)](#)

This constructor creates an instance of the OracleBFile class with an OracleConnection object.

- [OracleBFile\(OracleConnection, string, string\)](#)

This constructor creates an instance of the OracleBFile class with an OracleConnection object, the location of the BFILE, and the name of the BFILE.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

OracleBFile(OracleConnection)

This constructor creates an instance of the OracleBFile class with an OracleConnection object.

Declaration

```
// C#  
public OracleBFile(OracleConnection con);
```

Parameters

- *con*

The OracleConnection object.

Exceptions

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

The connection must be opened explicitly by the application. OracleBFile does not open the connection implicitly.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

OracleBFile(OracleConnection, string, string)

This constructor creates an instance of the `OracleBFile` class with an `OracleConnection` object, the location of the BFILE, and the name of the BFILE.

Declaration

```
// C#  
public OracleBFile(OracleConnection con, string directoryName, string  
    fileName);
```

Parameters

- *con*
The `OracleConnection` object.
- *directoryName*
The directory alias created by the `CREATE DIRECTORY` SQL statement.
- *fileName*
The name of the external LOB.

Exceptions

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The `OracleConnection` must be opened explicitly by the application. `OracleBFile` does not open the connection implicitly.

To initialize a BFILE column using an `OracleBFile` instance as an input parameter of a SQL INSERT statement, *directoryName* and *fileName* must be properly set.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

OracleBFile Static Fields

OracleBFile static fields are listed in [Table 13-6](#).

Table 13-6 OracleBFile Static Fields

Field	Description
MaxSize	The static field holds the maximum number of bytes a BFILE can hold, which is 4,294,967,295 ($2^{32} - 1$) bytes
Null	Represents a null value that can be assigned to the value of an OracleBFile instance

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleBFile Class](#)
- [OracleBFile Members](#)

MaxSize

This static field holds the maximum number of bytes a BFILE can hold, which is 4,294,967,295 ($2^{32} - 1$) bytes.

Declaration

```
// C#  
public static readonly Int64 MaxSize = 4294967295;
```

Remarks

This field is useful in code that checks whether or not the operation exceeds the maximum length allowed.

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleBFile Class](#)
- [OracleBFile Members](#)

Null

This static field represents a null value that can be assigned to the value of an OracleBFile instance.

Declaration

```
// C#  
public static readonly OracleBFile Null;
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

OracleBFile Static Methods

OracleBFile static methods are listed in [Table 13-7](#).

Table 13-7 OracleBFile Static Methods

Methods	Description
Equals	Inherited from System.Object (Overloaded)

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

OracleBFile Instance Properties

OracleBFile instance properties are listed in [Table 13-8](#).

Table 13-8 OracleBFile Instance Properties

Properties	Description
CanRead	Indicates whether or not the LOB stream can be read
CanSeek	Indicates whether or not forward and backward seek operations can be performed
CanWrite	Indicates whether or not the LOB object supports writing
Connection	Indicates the connection used to read from a BFILE
DirectoryName	Indicates the directory alias of the BFILE
FileExists	Indicates whether or not the specified BFILE exists
FileName	Indicates the name of the BFILE
IsEmpty	Indicates whether the BFILE is empty or not

Table 13-8 (Cont.) OracleBFile Instance Properties

Properties	Description
IsNull	Indicates whether or not the current instance has a null value
IsOpen	Indicates whether the BFILE has been opened by this instance or not
Length	Indicates the size of the BFILE data in bytes
Position	Indicates the current read position in the LOB stream
Value	Returns the data, starting from the first byte in BFILE, as a byte array

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleBFile Class](#)
- [OracleBFile Members](#)

CanRead

Overrides Stream

This instance property indicates whether or not the LOB stream can be read.

Declaration

```
// C#  
public override bool CanRead{get;}
```

Property Value

If the LOB stream can be read, returns true; otherwise, returns false.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleBFile Class](#)
- [OracleBFile Members](#)

CanSeek

Overrides Stream

This instance property indicates whether or not forward and backward seek operations can be performed.

Declaration

```
// C#  
public override bool CanSeek{get;}
```

Property Value

If forward and backward seek operations can be performed, returns true; otherwise, returns false.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

CanWrite

Overrides Stream

This instance property indicates whether or not the LOB object supports writing.

Declaration

```
// C#  
public override bool CanWrite{get;}
```

Property Value

BFILE is read only.

Remarks

BFILE is read-only, therefore, the boolean value is always false.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

Connection

This instance property indicates the connection used to read from a BFILE.

Declaration

```
// C#  
public OracleConnection Connection {get;}
```

Property Value

An object of `OracleConnection`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

DirectoryName

This instance property indicates the directory alias of the BFILE.

Declaration

```
// C#  
public string DirectoryName {get;set;}
```

Property Value

A string.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The value of the `DirectoryName` changed while the BFILE is open.

Remarks

The maximum length of a `DirectoryName` is 30 bytes.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

FileExists

This instance property indicates whether or not the BFILE specified by the `DirectoryName` and `FileName` exists.

Declaration

```
// C#  
public bool FileExists {get;}
```

Property Value

bool

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

Unless a connection, file name, and directory name are provided, this property is set to false by default.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

FileName

This instance property indicates the name of the BFILE.

Declaration

```
// C#  
public string FileName {get;set}
```

Property Value

A string that contains the BFILE name.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The value of the DirectoryName changed while the BFILE is open.

Remarks

The maximum length of a FileName is 255 bytes.

Changing the FileName property while the BFILE object is opened causes an exception.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

IsEmpty

This instance property indicates whether the BFILE is empty or not.

Declaration

```
// C#  
public bool IsEmpty {get;}
```

Property Value

bool

Exceptions

ObjectDisposedException - The object is already disposed.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

IsNull

This property indicates whether or not the current instance has a null value.

Declaration

```
// C#  
public bool IsNull{get;}
```

Property Value

Returns true if the current instance has a null value; otherwise, returns false.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

IsOpen

This instance property indicates whether the BFILE has been opened by this instance or not.

Declaration

```
// C#  
public bool IsOpen {get;}
```

Property Value

A bool.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

Length

Overrides Stream

This instance property indicates the size of the BFILE data in bytes.

Declaration

```
// C#  
public override Int64 Length {get;}
```

Property Value

Int64

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

Position

Overrides Stream

This instance property indicates the current read position in the LOB stream.

Declaration

```
// C#  
public override Int64 Position{get; set;}
```

Property Value

An Int64 value that indicates the read position.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

ArgumentOutOfRangeException - The value is less than 0.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

Value

This instance property returns the data, starting from the first byte in BFILE, as a byte array.

Declaration

```
// C#  
public byte[] Value{get;}
```

Property Value

A byte array.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

The length of data is bound by the maximum length of the byte array. The current value of the Position property is not used or changed.

For very large values, .NET may not be able to allocate enough contiguous memory to store all the data. Typically, you will see a System.OverflowException or System.OutOfMemoryException error then. To retrieve the large value, you can retrieve the data in chunks by calling GetChars() or GetBytes() method on the OracleDataReader. You can also call Read() method on the class.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

OracleBFile Instance Methods

OracleBFile instance methods are listed in [Table 13-9](#).

Table 13-9 OracleBFile Instance Methods

Methods	Description
BeginRead	Inherited from System.IO.Stream
BeginWrite	<i>Not Supported</i>
Clone	Creates a copy of an OracleBFile object
Close	Closes the current stream and releases any resources associated with the stream
CloseFile	Closes the BFILE referenced by the current BFILE instance
Compare	Compares data referenced by the two OracleBFiles
CreateObjRef	Inherited from System.MarshalByRefObject
CopyTo	Copies data as specified (Overloaded)
CopyToAsync	Returns a Task-based asynchronous version of OracleBFile.CopyTo(), which copies data from the current instance to the provided object. (Overloaded)
Dispose	Releases resources allocated by this object
EndRead	Inherited from System.IO.Stream
EndWrite	<i>Not Supported</i>

Table 13-9 (Cont.) OracleBFile Instance Methods

Methods	Description
Equals	Inherited from System.Object (Overloaded)
Flush	<i>Not Supported</i>
FlushAsync	<i>Not Supported</i>
GetHashCode	Inherited from System.Object
GetLifetimeService	Inherited from System.MarshalByRefObject
GetType	Inherited from System.Object
InitializeLifetimeService	Inherited from System.MarshalByRefObject
IsEqual	Compares the LOB references
OpenFile	Opens the BFILE specified by the FileName and DirectoryName
Read	Reads a specified amount of bytes from the OracleBFile instance and populates the buffer
ReadAsync	Returns a Task-based asynchronous version of OracleBFile.Read(), which reads a specified number of bytes from the OracleBFile instance and populates the buffer (Overloaded)
ReadByte	Inherited from System.IO.Stream
Search	Searches for a binary pattern in the current instance of an OracleBFile
Seek	Sets the position on the current LOB stream
SetLength	<i>Not Supported</i>
ToString	Inherited from System.Object
Write	<i>Not Supported</i>
WriteByte	<i>Not Supported</i>

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

Clone

This instance method creates a copy of an OracleBFile object.

Declaration

```
// C#
public object Clone();
```

Return Value

An OracleBFile object.

Implements

ICloneable

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

The cloned object has the same property values as that of the object being cloned.

Example

```
// Database Setup, if you have not done so yet.
/* Log on as DBA (SYS or SYSTEM) that has CREATE ANY DIRECTORY privilege.

CREATE OR REPLACE DIRECTORY MYDIR AS 'C:\TEMP';

*/

// C#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class CloneSample
{
    static void Main()
    {
        // Create MYDIR directory object as indicated above and create a file
        // MyFile.txt with the text ABCDABC under C:\TEMP directory.
        // Note that the byte representation of the ABCDABC is 65666768656667

        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleBFile bFile1 = new OracleBFile(con, "MYDIR", "MyFile.txt");

        // Open the OracleBFile
        bFile1.OpenFile();

        // Prints "bFile1.Position = 0"
        Console.WriteLine("bFile1.Position = " + bFile1.Position);

        // Set the Position before calling Clone()
        bFile1.Position = 1;

        // Clone the OracleBFile
        OracleBFile bFile2 = (OracleBFile) bFile1.Clone();

        // Open the OracleBFile
```

```
bFile2.OpenFile();

// Prints "bFile2.Position = 1"
Console.WriteLine("bFile2.Position = " + bFile2.Position);

// Close the OracleBFile
bFile1.CloseFile();

bFile1.Close();
bFile1.Dispose();

// Close the Cloned OracleBFile
bFile2.CloseFile();

bFile2.Close();
bFile2.Dispose();

con.Close();
con.Dispose();
}
}
```

📘 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

Close

Overrides Stream

This instance method closes the current stream and releases any resources associated with it.

Declaration

```
// C#
public override void Close();
```

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

📘 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

CloseFile

This instance method closes the BFILE referenced by the current BFILE instance.

Declaration

```
// C#  
public void CloseFile();
```

Remarks

No error is returned if the BFILE exists, but is not opened.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

Compare

This instance method compares data referenced by the two OracleBFiles.

Declaration

```
// C#  
public int Compare(Int64 src_offset, OracleBFile obj, Int64 dst_offset,  
    Int64 amount);
```

Parameters

- *src_offset*
The offset of the current instance.
- *obj*
The provided OracleBFile object.
- *dst_offset*
The offset of the OracleBFile object.
- *amount*
The number of bytes to compare.

Return Value

Returns a number that is:

- Less than zero: if the BFILE data of the current instance is less than that of the provided BFILE data.
- Zero: if both the BFILES store the same data.

- Greater than zero: if the BFILE data of the current instance is greater than that of the provided BFILE data.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - The `src_offset`, the `dst_offset`, or the `amount` is less than 0.

Remarks

The provided object and the current instance must be using the same connection, that is, the same `OracleConnection` object.

The BFILE needs to be opened using `OpenFile` before the operation.

Example

```
// Database Setup, if you have not done so yet.
/* Log on as DBA (SYS or SYSTEM) that has CREATE ANY DIRECTORY privilege.

CREATE OR REPLACE DIRECTORY MYDIR AS 'C:\TEMP';

*/

// C#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class CompareSample
{
    static void Main()
    {
        // Create MYDIR directory object as indicated previously and create a file
        // MyFile.txt with the text ABCDABC under C:\TEMP directory.
        // Note that the byte representation of the ABCDABC is 65666768656667

        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleBFile bFile1 = new OracleBFile(con, "MYDIR", "MyFile.txt");
        OracleBFile bFile2 = new OracleBFile(con, "MYDIR", "MyFile.txt");

        // Open the OracleBFiles
        bFile1.OpenFile();
        bFile2.OpenFile();

        // Compare 2 bytes from the 1st byte of bFile1 and
        // the 5th byte of bFile2 onwards
        int result = bFile1.Compare(1, bFile2, 5, 2);

        // Prints "result = 0" (Indicates the data is identical)
        Console.WriteLine("result = " + result);

        // Close the OracleBFiles
```

```
bFile1.CloseFile();  
bFile2.CloseFile();  
  
bFile1.Close();  
bFile1.Dispose();  
  
bFile2.Close();  
bFile2.Dispose();  
  
con.Close();  
con.Dispose();  
}  
}
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

CopyTo

CopyTo copies data from the current instance to the provided object.

Overload List:

- [CopyTo\(OracleBlob\)](#)
This instance method copies data from the current instance to the provided OracleBlob object.
- [CopyTo\(OracleBlob, Int64\)](#)
This instance method copies data from the current OracleBFile instance to the provided OracleBlob object with the specified destination offset.
- [CopyTo\(Int64, OracleBlob, Int64, Int64\)](#)
This instance method copies data from the current OracleBFile instance to the provided OracleBlob object with the specified source offset, destination offset, and character amounts.
- [CopyTo\(OracleClob\)](#)
This instance method copies data from the current OracleBFile instance to the provided OracleClob object.
- [CopyTo\(OracleClob, Int64\)](#)
This instance method copies data from the current OracleBFile instance to the provided OracleClob object with the specified destination offset.
- [CopyTo\(Int64, OracleClob, Int64, Int64\)](#)
This instance method copies data from the current OracleBFile instance to the provided OracleClob object with the specified source offset, destination offset, and amount of characters.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

CopyTo(OracleBlob)

This instance method copies data from the current instance to the provided `OracleBlob` object.

Declaration

```
// C#  
public Int64 CopyTo(OracleBlob obj);
```

Parameters

- *obj*

The `OracleBlob` object to which the data is copied.

Return Value

The return value is the amount copied.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - This exception is thrown if any of the following conditions exist:

- The `OracleConnection` is not open or has been closed during the lifetime of the object.
- The LOB object parameter has a different connection than the object.

Remarks

The provided object and the current instance must be using the same connection; that is, the same `OracleConnection` object.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

CopyTo(OracleBlob, Int64)

This instance method copies data from the current `OracleBFile` instance to the provided `OracleBlob` object with the specified destination offset.

Declaration

```
// C#  
public Int64 CopyTo(OracleBlob obj, Int64 dst_offset);
```

Parameters

- *obj*
The OracleBlob object to which the data is copied.
- *dst_offset*
The offset (in bytes) at which the OracleBlob object is copied.

Return Value

The return value is the amount copied.

Exceptions

ObjectDisposedException - The object is already disposed.

ArgumentOutOfRangeException - The *dst_offset* is less than 0.

InvalidOperationException - This exception is thrown if any of the following conditions exist:

- The OracleConnection is not open or has been closed during the lifetime of the object.
- The LOB object parameter has a different connection than the object.

Remarks

If the *dst_offset* is beyond the end of the OracleBlob data, spaces are written into the OracleBlob until the *dst_offset* is met.

The offsets are 0-based. No character conversion is performed by this operation.

The provided object and the current instance must be using the same connection; that is, the same OracleConnection object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

CopyTo(Int64, OracleBlob, Int64, Int64)

This instance method copies data from the current OracleBFile instance to the provided OracleBlob object with the specified source offset, destination offset, and character amounts.

Declaration

```
// C#  
public Int64 CopyTo(Int64 src_offset, OracleBlob obj, Int64 dst_offset,  
    Int64 amount);
```

Parameters

- *src_offset*
The offset (in bytes) in the current instance, from which the data is read.
- *obj*
An OracleBlob object to which the data is copied.
- *dst_offset*
The offset (in bytes) to which the OracleBlob object is copied.
- *amount*
The amount of data to be copied.

Return Value

The return value is the amount copied.

Exceptions

ObjectDisposedException - The object is already disposed.

ArgumentOutOfRangeException - The *src_offset*, the *dst_offset*, or the *amount* is less than 0.

InvalidOperationException - This exception is thrown if any of the following conditions exist:

- The OracleConnection is not open or has been closed during the lifetime of the object.
- The LOB object parameter has a different connection than the object.

Remarks

If the *dst_offset* is beyond the end of the OracleBlob data, spaces are written into the OracleBlob until the *dst_offset* is met.

The offsets are 0-based. No character conversion is performed by this operation.

The provided object and the current instance must be using the same connection; that is, the same OracleConnection object.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

CopyTo(OracleClob)

This instance method copies data from the current OracleBFile instance to the provided OracleClob object.

Declaration

```
// C#  
public Int64 CopyTo(OracleClob obj);
```

Parameters

- *obj*

The OracleClob object to which the data is copied.

Return Value

The return value is the amount copied.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - This exception is thrown if any of the following conditions exist:

- The OracleConnection is not open or has been closed during the lifetime of the object.
- The LOB object parameter has a different connection than the object.

Remarks

The provided object and the current instance must be using the same connection, that is, the same OracleConnection object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

CopyTo(OracleClob, Int64)

This instance method copies data from the current OracleBFile instance to the provided OracleClob object with the specified destination offset.

Declaration

```
// C#  
public Int64 CopyTo(OracleClob obj, Int64 dst_offset);
```

Parameters

- *obj*

The OracleClob object that the data is copied to.

- *dst_offset*

The offset (in characters) at which the OracleClob object is copied to.

Return Value

The amount copied.

Exceptions

Exceptions

ObjectDisposedException - The object is already disposed.

ArgumentOutOfRangeException - The *dst_offset* is less than 0.

InvalidOperationException - This exception is thrown if any of the following conditions exist:

- The OracleConnection is not open or has been closed during the lifetime of the object.
- The LOB object parameter has a different connection than the object.

Remarks

If the *dst_offset* is beyond the end of the OracleClob data, spaces are written into the OracleClob until the *dst_offset* is met.

The offsets are 0-based. No character conversion is performed by this operation.

The provided object and the current instance must be using the same connection, that is, the same OracleConnection object.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

CopyTo(Int64, OracleClob, Int64, Int64)

This instance method copies data from the current OracleBFile instance to the provided OracleClob object with the specified source offset, destination offset, and amount of characters.

Declaration

```
// C#
public Int64 CopyTo(Int64 src_offset, OracleClob obj, Int64 dst_offset,
    Int64 amount);
```

Parameters

- *src_offset*
The offset (in characters) in the current instance, from which the data is read.
- *obj*
An OracleClob object that the data is copied to.
- *dst_offset*

The offset (in characters) at which the `OracleClob` object is copied to.

- *amount*

The amount of data to be copied.

Return Value

The return value is the amount copied.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentOutOfRangeException` - The *src_offset*, the *dst_offset*, or the *amount* is less than 0.

`InvalidOperationException` - This exception is thrown if any of the following conditions exist:

- The `OracleConnection` is not open or has been closed during the lifetime of the object.
- The LOB object parameter has a different connection than the object.

Remarks

If the *dst_offset* is beyond the end of the current `OracleClob` data, spaces are written into the `OracleClob` until the *dst_offset* is met.

The offsets are 0-based. No character conversion is performed by this operation.

The provided object and the current instance must be using the same connection, that is, the same `OracleConnection` object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

CopyToAsync

`CopyToAsync` returns a Task-based asynchronous version of `OracleBFile.CopyTo()`, which copies data from the current instance to the provided object.

Overload List:

- [CopyToAsync\(Int64, OracleBlob, Int64, Int64\)](#)

This method returns a Task-based asynchronous version of `OracleBFile.CopyTo()`, which copies data from the current instance to the provided object.

- [CopyToAsync\(Int64, OracleBlob, Int64, Int64, CancellationToken\)](#)

`CopyToAsync` returns a Task-based asynchronous version of `OracleBFile.CopyTo()`, which copies data from the current instance to the provided object.

- [CopyToAsync\(OracleBlob, Int64\)](#)

CopyToAsync returns a Task-based asynchronous version of `OracleBFile.CopyTo()`, which copies data from the current instance to the provided object.

- [CopyToAsync\(OracleBlob, Int64, CancellationToken\)](#)

CopyToAsync returns a Task-based asynchronous version of `OracleBFile.CopyTo()`, which copies data from the current instance to the provided object.

- [CopyToAsync\(OracleBlob\)](#)

CopyToAsync returns a Task-based asynchronous version of `OracleBFile.CopyTo()`, which copies data from the current instance to the provided object.

- [CopyToAsync\(OracleBlob, CancellationToken\)](#)

CopyToAsync returns a Task-based asynchronous version of `OracleBFile.CopyTo()`, which copies data from the current instance to the provided object.

- [CopyToAsync\(Int64, OracleClob, Int64, Int64\)](#)

This method returns a Task-based asynchronous version of `OracleBFile.CopyTo()`, which copies data from the current instance to the provided object.

- [CopyToAsync\(Int64, OracleClob, Int64, Int64, CancellationToken\)](#)

This method returns a Task-based asynchronous version of `OracleBFile.CopyTo()`, which copies data from the current instance to the provided object.

- [CopyToAsync\(OracleClob, Int64\)](#)

This method returns a Task-based asynchronous version of `OracleBFile.CopyTo()`, which copies data from the current instance to the provided object.

- [CopyToAsync\(OracleClob, Int64, CancellationToken\)](#)

This method returns a Task-based asynchronous version of `OracleBFile.CopyTo()`, which copies data from the current instance to the provided object.

- [CopyToAsync\(OracleClob\)](#)

This method returns a Task-based asynchronous version of `OracleBFile.CopyTo()`, which copies data from the current instance to the provided object.

- [CopyToAsync\(OracleClob, CancellationToken\)](#)

This method returns a Task-based asynchronous version of `OracleBFile.CopyTo()`, which copies data from the current instance to the provided object.

Example (includes all overloads)

```
using Oracle.ManagedDataAccess.Client;
using Oracle.ManagedDataAccess.Types;
using System;
using System.Threading;
using System.Threading.Tasks;

namespace AsyncApp
{
    class AsyncDemo
    {
        static async Task Main(string[] args)
        {
            string connectionString = "User Id=HR; Password=<PASSWORD>; Data Source=oracle;";

            OracleConnection oc = new OracleConnection(connectionString);
            await oc.OpenAsync(Cancellation_token.None);
        }
    }
}
```

```
OracleCommand cmd = oc.CreateCommand();
cmd.CommandText = " select bfile_column from tab1";

OracleBlob blob1 = new OracleBlob(oc);
OracleBlob blob2 = new OracleBlob(oc);
OracleBlob blob3 = new OracleBlob(oc);
OracleBlob blob4 = new OracleBlob(oc);
OracleBlob blob5 = new OracleBlob(oc);
OracleBlob blob6 = new OracleBlob(oc);

OracleClob clob1 = new OracleClob(oc);
OracleClob clob2 = new OracleClob(oc);
OracleClob clob3 = new OracleClob(oc);
OracleClob clob4 = new OracleClob(oc);
OracleClob clob5 = new OracleClob(oc);
OracleClob clob6 = new OracleClob(oc);

OracleDataReader reader = await cmd.ExecuteReaderAsync();

await reader.ReadAsync(CancellationTokens.None);
using (OracleBFile bfile = reader.GetOracleBFile(0))
{
    //asynchronously copy bfile data
    Int64 bytesCopied1 = await bfile.CopyToAsync(0, blob1, 0, bfile.Length);
    Console.WriteLine("bytes copied to blob1 = " + bytesCopied1);

    //asynchronously copy bfile data
    Int64 bytesCopied2 = await bfile.CopyToAsync(0, blob2, 0, bfile.Length,
CancellationTokens.None);
    Console.WriteLine("bytes copied to blob2 = " + bytesCopied2);

    //asynchronously copy bfile data
    Int64 bytesCopied3 = await bfile.CopyToAsync(blob3, 0);
    Console.WriteLine("bytes copied to blob3 = " + bytesCopied3);

    //asynchronously copy bfile data
    Int64 bytesCopied4 = await bfile.CopyToAsync(blob4, 0, CancellationTokens.None);
    Console.WriteLine("bytes copied to blob4 = " + bytesCopied4);

    //asynchronously copy bfile data
    Int64 bytesCopied5 = await bfile.CopyToAsync(blob5);
    Console.WriteLine("bytes copied to blob5 = " + bytesCopied5);

    //asynchronously copy bfile data
    Int64 bytesCopied6 = await bfile.CopyToAsync(blob6, CancellationTokens.None);
    Console.WriteLine("bytes copied to blob6 = " + bytesCopied6);

    //asynchronously copy bfile data
    bytesCopied1 = await bfile.CopyToAsync(0, clob1, 0, bfile.Length);
    Console.WriteLine("bytes copied to clob1 = " + bytesCopied1);

    //asynchronously copy bfile data
    bytesCopied2 = await bfile.CopyToAsync(0, clob2, 0, bfile.Length,
CancellationTokens.None);
    Console.WriteLine("bytes copied to clob2 = " + bytesCopied2);

    //asynchronously copy bfile data
    bytesCopied3 = await bfile.CopyToAsync(clob3, 0);
    Console.WriteLine("bytes copied to clob3 = " + bytesCopied3);

    //asynchronously copy bfile data
```

```
bytesCopied4 = await bfile.CopyToAsync(clob4, 0, CancellationToken.None);
Console.WriteLine("bytes copied to clob4 = " + bytesCopied4);

//asynchronously copy bfile data
bytesCopied5 = await bfile.CopyToAsync(clob5);
Console.WriteLine("bytes copied to clob5 = " + bytesCopied5);

//asynchronously copy bfile data
bytesCopied6 = await bfile.CopyToAsync(clob6, CancellationToken.None);
Console.WriteLine("bytes copied to clob6 = " + bytesCopied6);
    }
}
}
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

CopyToAsync(Int64, OracleBlob, Int64, Int64)

This method returns a Task-based asynchronous version of `OracleBFile.CopyTo()`, which copies data from the current instance to the provided object.

Declaration

```
// C#
public Task<Int64> CopyToAsync(Int64 src_offset, OracleBlob obj, Int64 dst_offset, Int64
amount)
```

Parameters

- `src_offset` - The offset (in bytes) in the current instance, from which the data is read.
- `obj` - The `OracleBlob` object to which the data is copied.
- `dst_offset` - The offset (in bytes) at which the `OracleBlob` object is copied.
- `amount` - The amount of data to be copied.

Return Value

`Task<Int64>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `InvalidOperationException` - The parameter has a different connection than the object, `OracleConnection` is not opened, or `OracleConnection` has been reopened.
- `ArgumentOutOfRangeException` - The `src_offset`, the `dst_offset`, or the amount parameter is less than 0.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

CopyToAsync(Int64, OracleBlob, Int64, Int64, CancellationToken)

This method returns a Task-based asynchronous version of `OracleBFile.CopyTo()`, which copies data from the current instance to the provided object.

Declaration

```
// C#
public Task<Int64> CopyToAsync(Int64 src_offset, OracleBlob obj, Int64 dst_offset, Int64
amount, CancellationToken cancellation_token)
```

Parameters

- `src_offset` - The offset (in bytes) in the current instance, from which the data is read.
- `obj` - The `OracleBlob` object to which the data is copied.
- `dst_offset` - The offset (in bytes) at which the `OracleBlob` object is copied.
- `amount` - The amount of data to be copied.
- `cancellation_token` - The input cancellation token which can be used by the application to cancel the task.

Return Value

`Task<Int64>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `ArgumentOutOfRangeException` - The `src_offset`, the `dst_offset`, or the `amount` parameter is less than 0.
- `InvalidOperationException` - This exception is thrown if any of the following conditions exist:
 - The `OracleConnection` is not open or has been closed during the lifetime of the object.
 - The LOB object parameter has a different connection than the object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

CopyToAsync(OracleBlob, Int64)

This method returns a Task-based asynchronous version of `OracleBFile.CopyTo()`, which copies data from the current instance to the provided object.

Declaration

```
// C#  
public async Task<Int64> CopyToAsync(OracleBlob obj, Int64 dst_offset)
```

Parameters

- `obj` - The `OracleBlob` object to which the data is copied.
- `dst_offset` - The offset (in bytes) at which the `OracleBlob` object is copied.

Return Value

`Task<Int64>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `ArgumentOutOfRangeException` - The `dst_offset` parameter is less than 0.
- `InvalidOperationException` - This exception is thrown if any of the following conditions exist:
 - The `OracleConnection` is not open or has been closed during the lifetime of the object.
 - The LOB object parameter has a different connection than the object.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

CopyToAsync(OracleBlob, Int64, CancellationToken)

This method returns a Task-based asynchronous version of `OracleBFile.CopyTo()`, which copies data from the current instance to the provided object.

Declaration

```
// C#  
public async Task<Int64> CopyToAsync(OracleBlob obj, Int64 dst_offset, CancellationToken  
cancellation_token)
```

Parameters

- `obj` - The `OracleBlob` object to which the data is copied.
- `dst_offset` - The offset (in bytes) at which the `OracleBlob` object is copied.

- `cancellationToken` - The input cancellation token which can be used by the application to cancel the task.

Return Value

`Task<Int64>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `ArgumentOutOfRangeException` - The `dst_offset` is less than 0.
- `InvalidOperationException` - This exception is thrown if any of the following conditions exist:
 - The `OracleConnection` is not open or has been closed during the lifetime of the object.
 - The LOB object parameter has a different connection than the object.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

CopyToAsync(OracleBlob)

This method returns a Task-based asynchronous version of `OracleBFile.CopyTo()`, which copies data from the current instance to the provided object.

Declaration

```
// C#  
public async Task<Int64> CopyToAsync(OracleBlob obj)
```

Parameters

- `obj` - The `OracleBlob` object to which the data is copied.

Return Value

`Task<Int64>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `InvalidOperationException` - This exception is thrown if any of the following conditions exist:
 - The `OracleConnection` is not open or has been closed during the lifetime of the object.
 - The LOB object parameter has a different connection than the object.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

CopyToAsync(OracleBlob, CancellationToken)

This method returns a Task-based asynchronous version of `OracleBFile.CopyTo()`, which copies data from the current instance to the provided object.

Declaration

```
// C#  
public async Task<Int64> CopyToAsync(OracleBlob obj, CancellationToken cancellationToken)
```

Parameters

- `obj` - The `OracleBlob` object to which the data is copied.
- `cancellationToken` - The input cancellation token which can be used by the application to cancel the task.

Return Value

`Task<Int64>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `InvalidOperationException` - This exception is thrown if any of the following conditions exist:
 - The `OracleConnection` is not open or has been closed during the lifetime of the object.
 - The LOB object parameter has a different connection than the object.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

CopyToAsync(Int64, OracleClob, Int64, Int64)

This method returns a Task-based asynchronous version of `OracleBFile.CopyTo()`, which copies data from the current instance to the provided object.

Declaration

```
// C#  
public Task<Int64> CopyToAsync(Int64 src_offset, OracleClob obj, Int64 dst_offset, Int64  
amount)
```

Parameters

- `src_offset` - The offset (in characters) in the current instance, from which the data is read.
- `obj` - The `OracleClob` object to which the data is copied.
- `dst_offset` - The offset (in characters) at which the `OracleClob` object is copied.
- `amount` - The amount of data to be copied.

Return Value

`Task<Int64>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `ArgumentOutOfRangeException` - The `src_offset`, the `dst_offset`, or the `amount` parameter is less than 0.
- `InvalidOperationException` - The parameter has a different connection than the object, `OracleConnection` is not opened, or `OracleConnection` has been reopened.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

CopyToAsync(Int64, OracleClob, Int64, Int64, CancellationToken)

This method returns a Task-based asynchronous version of `OracleBFile.CopyTo()`, which copies data from the current instance to the provided object.

Declaration

```
// C#  
public Task<Int64> CopyToAsync(Int64 src_offset, OracleClob obj, Int64 dst_offset, Int64  
amount, CancellationToken cancellationToken)
```

Parameters

- `src_offset` - The offset (in characters) in the current instance, from which the data is read.
- `obj` - The `OracleClob` object to which the data is copied.
- `dst_offset` - The offset (in characters) at which the `OracleClob` object is copied.

- `amount` - The amount of data to be copied.
- `cancellationToken` - The input cancellation token which can be used by the application to cancel the task.

Return Value

`Task<Int64>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `ArgumentOutOfRangeException` - The `src_offset`, the `dst_offset`, or the `amount` parameter is less than 0.
- `InvalidOperationException` - This exception is thrown if any of the following conditions exist:
 - The `OracleConnection` is not open or has been closed during the lifetime of the object.
 - The `LOB` object parameter has a different connection than the object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

CopyToAsync(OracleClob, Int64)

This method returns a Task-based asynchronous version of `OracleBFile.CopyTo()`, which copies data from the current instance to the provided object.

Declaration

```
// C#  
public async Task<Int64> CopyToAsync(OracleClob obj, Int64 dst_offset)
```

Parameters

- `obj` - The `OracleClob` object to which the data is copied.
- `dst_offset` - The offset (in characters) at which the `OracleClob` object is copied.

Return Value

`Task<Int64>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `ArgumentOutOfRangeException` - The `dst_offset` parameter is less than 0.

- `InvalidOperationException` - This exception is thrown if any of the following conditions exist:
 - The `OracleConnection` is not open or has been closed during the lifetime of the object.
 - The LOB object parameter has a different connection than the object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

CopyToAsync(OracleClob, Int64, CancellationToken)

This method returns a Task-based asynchronous version of `OracleBFile.CopyTo()`, which copies data from the current instance to the provided object.

Declaration

```
// C#  
public async Task<Int64> CopyToAsync(OracleClob obj, Int64 dst_offset, CancellationToken  
cancellation_token)
```

Parameters

- `obj` - The `OracleClob` object to which the data is copied.
- `dst_offset` - The offset (in characters) at which the `OracleClob` object is copied.
- `cancellation_token` - The input cancellation token which can be used by the application to cancel the task.

Return Value

`Task<Int64>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `ArgumentOutOfRangeException` - The `dst_offset` is less than 0.
- `InvalidOperationException` - This exception is thrown if any of the following conditions exist:
 - The `OracleConnection` is not open or has been closed during the lifetime of the object.
 - The LOB object parameter has a different connection than the object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

CopyToAsync(OracleClob)

This method returns a Task-based asynchronous version of `OracleBFile.CopyTo()`, which copies data from the current instance to the provided object.

Declaration

```
// C#  
public async Task<Int64> CopyToAsync(OracleClob obj)
```

Parameters

- `obj` - The `OracleClob` object to which the data is copied.

Return Value

`Task<Int64>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `InvalidOperationException` - This exception is thrown if any of the following conditions exist:
 - The `OracleConnection` is not open or has been closed during the lifetime of the object.
 - The LOB object parameter has a different connection than the object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

CopyToAsync(OracleClob, CancellationToken)

This method returns a Task-based asynchronous version of `OracleBFile.CopyTo()`, which copies data from the current instance to the provided object.

Declaration

```
// C#  
public async Task<Int64> CopyToAsync(OracleClob obj, CancellationToken cancellationToken)
```


Parameters

- `obj` - The `OracleClob` object to which the data is copied.
- `cancellationToken` - The input cancellation token which can be used by the application to cancel the task.

Return Value

`Task<Int64>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `InvalidOperationException` - This exception is thrown if any of the following conditions exist:
 - The `OracleConnection` is not open or has been closed during the lifetime of the object.
 - The LOB object parameter has a different connection than the object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

Dispose

This instance method releases resources allocated by this object.

Declaration

```
// C#  
public void Dispose();
```

Implements

`IDisposable`

Remarks

Although some properties can still be accessed, their values may not be accountable. Since resources are freed, method calls may lead to exceptions. The object cannot be reused after being disposed.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

Flush

This method is not supported.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

FlushAsync

This method is not supported.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

IsEqual

This instance method compares the LOB references.

Declaration

```
// C#  
public bool IsEqual(OracleBFile obj);
```

Parameters

- *obj*
The provided OracleBFile object.

Return Value

Returns true if the current OracleBFile and the provided OracleBFile object refer to the same external LOB. Returns false otherwise.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

Note that this method can return true even if the two OracleBFile objects return false for == or Equals() since two different OracleBFile instances can refer to the same external LOB.

The provided object and the current instance must be using the same connection; that is, the same OracleConnection object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

OpenFile

This instance method opens the BFILE specified by the FileName and DirectoryName.

Declaration

```
// C#  
public void OpenFile();
```

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

Many operations, such as Compare(), CopyTo(), Read(), and Search() require that the BFILE be opened using OpenFile before the operation.

Calling OpenFile on an opened BFILE is not operational.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

Read

Overrides Stream

This instance method reads a specified amount of bytes from the `OracleBFile` instance and populates the buffer.

Declaration

```
// C#  
public override int Read(byte[] buffer, int offset, int count);
```

Parameters

- *buffer*
The byte array buffer to be populated.
- *offset*
The offset of the byte array buffer to be populated.
- *count*
The amount of bytes to read.

Return Value

The return value indicates the number of bytes read from the BFILE, that is, the external LOB.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - Either the *offset* or the *count* parameter is less than 0 or the *offset* is greater than or equal to the *buffer.Length* or the *offset* and the *count* together are greater than *buffer.Length*.

Remarks

The LOB data is read starting from the position specified by the `Position` property.

Example

```
// Database Setup, if you have not done so yet.  
/* Log on as DBA (SYS or SYSTEM) that has CREATE ANY DIRECTORY privilege.  
  
CREATE OR REPLACE DIRECTORY MYDIR AS 'C:\TEMP';  
  
*/  
  
// C#  
  
using System;  
using Oracle.DataAccess.Client;  
using Oracle.DataAccess.Types;  
  
class ReadSample  
{
```

```
static void Main()
{
    // Create MYDIR directory object as indicated previously and create a file
    // MyFile.txt with the text ABCDABC under C:\TEMP directory.
    // Note that the byte representation of the ABCDABC is 65666768656667

    string constr = "User Id=scott;Password=tiger;Data Source=oracle";
    OracleConnection con = new OracleConnection(constr);
    con.Open();

    OracleBFile bFile = new OracleBFile(con, "MYDIR", "MyFile.txt");

    // Open the OracleBFile
    bFile.OpenFile();

    // Read 7 bytes into readBuffer, starting at buffer offset 0
    byte[] readBuffer = new byte[7];
    int bytesRead = bFile.Read(readBuffer, 0, 7);

    // Prints "bytesRead = 7"
    Console.WriteLine("bytesRead = " + bytesRead);

    // Prints "readBuffer = 65666768656667"
    Console.Write("readBuffer = ");
    for(int index = 0; index < readBuffer.Length; index++)
    {
        Console.Write(readBuffer[index]);
    }
    Console.WriteLine();

    // Close the OracleBFile
    bFile.CloseFile();

    bFile.Close();
    bFile.Dispose();

    con.Close();
    con.Dispose();
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

ReadAsync

ReadAsync returns a Task-based asynchronous version of `OracleBFile.Read()`, which reads a specified number of bytes from the `OracleBFile` instance and populates the buffer.

Overload List:

- [ReadAsync\(byte\[\], int, int\)](#)

This method returns a Task-based asynchronous version of `OracleBFile.Read()`, which reads a specified number of bytes from the `OracleBFile` instance and populates the buffer.

- [ReadAsync\(byte\[\], int, int, CancellationToken\)](#)

This method returns a Task-based asynchronous version of `OracleBFile.Read()`, which reads a specified number of bytes from the `OracleBFile` instance and populates the buffer.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

ReadAsync(byte[], int, int)

This method returns a Task-based asynchronous version of `OracleBFile.Read()`, which reads a specified number of bytes from the `OracleBFile` instance and populates the buffer.

Declaration

```
// C#  
public Task<int> ReadAsync(byte[] buffer, int offset, int count)
```

Parameters

- `buffer` - The byte array buffer to be populated.
- `offset` - The starting offset (in bytes) at which the buffer is populated.
- `count` - The number of bytes to read.

Return Value

`Task<Int>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Implements

Stream

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.
- `ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:
 - The `offset` or the `count` parameter is less than 0.
 - The `offset` is greater than or equal to the `buffer.Length`.
 - The `offset` and the `count` together are greater than the `buffer.Length`.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

ReadAsync(byte[], int, int, CancellationToken)

This method returns a Task-based asynchronous version of `OracleBFile.Read()`, which reads a specified number of bytes from the `OracleBFile` instance and populates the buffer.

Declaration

```
// C#  
public override Task<int> ReadAsync(byte[] buffer, int offset, int count,  
CancellationToken cancellationToken)
```

Parameters

- `buffer` - The byte array buffer to be populated.
- `offset` - The starting offset (in bytes) at which the buffer is populated.
- `count` - The number of bytes to read.
- `cancellationToken` - The input cancellation token which can be used by the application to cancel the task.

Return Value

`Task<Int>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Implements

Stream

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.
- `ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:
 - The offset or the count parameter is less than 0.
 - The offset is greater than or equal to the `buffer.Length`.
 - The offset and the count together are greater than the `buffer.Length`.

Example

```
using Oracle.ManagedDataAccess.Client;  
using Oracle.ManagedDataAccess.Types;  
using System;
```

```

using System.Threading;
using System.Threading.Tasks;

namespace AsyncApp
{
    class AsyncDemo
    {
        static async Task Main(string[] args)
        {
            string connectionString = "User Id=HR; Password=<PASSWORD>; Data Source=oracle;";

            OracleConnection oc = new OracleConnection(connectionString);
            await oc.OpenAsync(CancellationToken.None);

            OracleCommand cmd = oc.CreateCommand();
            cmd.CommandText = " select bfile_column from tab1";

            OracleDataReader reader = await cmd.ExecuteReaderAsync();

            await reader.ReadAsync(CancellationToken.None);
            using (var bfile = reader.GetOracleBFile(0))
            {
                bfile.OpenFile();
                byte[] buffer = new byte[100];

                //asynchronously read bfile data
                Task<int> task = bfile.ReadAsync(buffer, 0, 64, CancellationToken.None);

                //other operations
                Console.WriteLine("Hello World");

                //await for asynchronous ReadAsync
                int bytesRead = await task;
                Console.WriteLine("Bytes Read: " + bytesRead);
            }
        }
    }
}

```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

Search

This instance method searches for a binary pattern in the current instance of an OracleBFile.

Declaration

```

// C#
public int Search(byte[] val, Int64 offset, Int64 nth);

```

Parameters

- *val*

The binary pattern being searched for.

- *offset*

The 0-based offset (in bytes) starting from which the OracleBFile is searched.

- *nth*

The specific occurrence (1-based) of the match for which the offset is returned.

Return Value

Returns the absolute *offset* of the start of the matched pattern (in bytes) for the *nth* occurrence of the match. Otherwise, 0 is returned.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

ArgumentOutOfRangeException - Either the *offset* is less than 0 or *nth* is less than or equal to 0 or *val.Length* is greater than 16383 or *nth* is greater than or equal to OracleBFile.MaxSize or *offset* is greater than or equal to OracleBFile.MaxSize.

Remarks

The limit of the search pattern is 16383 bytes.

Example

```
// Database Setup, if you have not done so yet.
/* Log on as DBA (SYS or SYSTEM) that has CREATE ANY DIRECTORY privilege.

CREATE OR REPLACE DIRECTORY MYDIR AS 'C:\TEMP';

*/

// C#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class SearchSample
{
    static void Main()
    {
        // Create MYDIR directory object as indicated previously and create a file
        // MyFile.txt with the text ABCDABC under C:\TEMP directory.
        // Note that the byte representation of the ABCDABC is 65666768656667

        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleBFile bFile = new OracleBFile(con, "MYDIR", "MyFile.txt");

        // Open the OracleBFile
        bFile.OpenFile();

        // Search for the 2nd occurrence of a byte pattern {66,67}
```

```
// starting from byte offset 1 in the OracleBFile
byte[] pattern = new byte[2] {66, 67};
long posFound = bFile.Search(pattern, 1, 2);

// Prints "posFound = 6"
Console.WriteLine("posFound = " + posFound);

// Close the OracleBFile
bFile.CloseFile();

bFile.Close();
bFile.Dispose();

con.Close();
con.Dispose();
}
}
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

Seek

Overrides Stream

This instance method sets the position on the current LOB stream.

Declaration

```
// C#
public override Int64 Seek(Int64 offset, SeekOrigin origin);
```

Parameters

- *offset*
A byte offset relative to origin.
- *origin*
A value of type `System.IO.SeekOrigin` indicating the reference point used to obtain the new position.

Return Value

Returns an `Int64` that indicates the position.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

If *offset* is negative, the new position precedes the position specified by *origin* by the number of bytes specified by *offset*.

If *offset* is zero, the new position is the position specified by *origin*.

If *offset* is positive, the new position follows the position specified by *origin* by the number of bytes specified by *offset*.

SeekOrigin.Begin specifies the beginning of a stream.

SeekOrigin.Current specifies the current position within a stream.

SeekOrigin.End specifies the end of a stream.

Example

```
// Database Setup, if you have not done so yet.
/* Log on as DBA (SYS or SYSTEM) that has CREATE ANY DIRECTORY privilege.

CREATE OR REPLACE DIRECTORY MYDIR AS 'C:\TEMP';

*/

// C#

using System;
using System.IO;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class SeekSample
{
    static void Main()
    {
        // Create MYDIR directory object as indicated previously and create a file
        // MyFile.txt with the text ABCDABC under C:\TEMP directory.
        // Note that the byte representation of the ABCDABC is 65666768656667

        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleBFile bFile = new OracleBFile(con, "MYDIR", "MyFile.txt");

        // Open the OracleBFile
        bFile.OpenFile();

        // Set the Position to 2 with respect to SeekOrigin.Begin
        long newPosition = bFile.Seek(2, SeekOrigin.Begin);

        // Prints "newPosition      = 2"
        Console.WriteLine("newPosition      = " + newPosition);

        // Prints "bFile.Position  = 2"
        Console.WriteLine("bFile.Position  = " + bFile.Position);

        // Read 2 bytes into readBuffer, starting at buffer offset 1
        byte[] readBuffer = new byte[4];
        int bytesRead = bFile.Read(readBuffer, 1, 2);
```

```
// Prints "bytesRead      = 2"
Console.WriteLine("bytesRead      = " + bytesRead);

// Prints "readBuffer      = 067680"
Console.Write("readBuffer      = ");
for(int index = 0; index < readBuffer.Length; index++)
{
    Console.Write(readBuffer[index]);
}
Console.WriteLine();

// Close the OracleBFile
bFile.CloseFile();

bFile.Close();
bFile.Dispose();

con.Close();
con.Dispose();
}
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

SetLength

This method is not supported.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

Write

This method is not supported.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBFile Class](#)
- [OracleBFile Members](#)

OracleBlob Class

An OracleBlob object is an object that has a reference to BLOB data. It provides methods for performing operations on BLOBs.

Class Inheritance

System.Object
 System.MarshalByRefObject
 System.IO.Stream
 Oracle.DataAccess.Types.OracleBlob

Declaration

```
// C#
public sealed class OracleBlob : Stream, ICloneable, INullable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class OracleBlobSample
```

```
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleBlob blob = new OracleBlob(con);

        // Write 4 bytes from writeBuffer, starting at buffer offset 0
        byte[] writeBuffer = new byte[4] {1, 2, 3, 4};
        blob.Write(writeBuffer, 0, 4);

        // Append first 2 bytes from writeBuffer {1, 2} to the oracleBlob
        blob.Append(writeBuffer, 0, 2);

        // Prints "blob.Length = 6"
        Console.WriteLine("blob.Length = " + blob.Length);

        // Reset the Position for the Read
        blob.Position = 0;

        // Read 6 bytes into readBuffer, starting at buffer offset 0
        byte[] readBuffer = new byte[6];
        int bytesRead = blob.Read(readBuffer, 0, 6);

        // Prints "bytesRead = 6"
        Console.WriteLine("bytesRead = " + bytesRead);

        // Prints "readBuffer = 123412"
        Console.Write("readBuffer = ");
        for(int index = 0; index < readBuffer.Length; index++)
        {
            Console.Write(readBuffer[index]);
        }
        Console.WriteLine();

        // Search for the 2nd occurrence of a byte pattern '12'
        // starting from byte offset 0 in the OracleBlob
        byte[] pattern = new byte[2] {1, 2};
        long posFound = blob.Search(pattern, 0, 2);

        // Prints "posFound = 5"
        Console.WriteLine("posFound = " + posFound);

        // Erase 4 bytes of data starting at byte offset 1
        // Sets bytes to zero
        blob.Erase(1, 4);

        byte[] erasedBuffer = blob.Value;

        //Prints "erasedBuffer = 100002"
        Console.Write("erasedBuffer = ");
        for(int index = 0; index < erasedBuffer.Length; index++)
        {
            Console.Write(erasedBuffer[index]);
        }
        Console.WriteLine();

        blob.Close();
        blob.Dispose();
    }
}
```

```
        con.Close();  
        con.Dispose();  
    }  
}
```

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleBlob Members](#)
- [OracleBlob Constructors](#)
- [OracleBlob Static Fields](#)
- [OracleBlob Static Methods](#)
- [OracleBlob Instance Properties](#)
- [OracleBlob Instance Methods](#)

OracleBlob Members

OracleBlob members are listed in the following tables.

OracleBlob Constructors

OracleBlob constructors are listed in [Table 13-10](#).

Table 13-10 OracleBlob Constructors

Constructor	Description
OracleBlob Constructors	Creates an instance of the OracleBlob class (Overloaded)

OracleBlob Static Fields

OracleBlob static fields are listed in [Table 13-11](#).

Table 13-11 OracleBlob Static Fields

Field	Description
MaxSize	Holds the maximum number of bytes a BLOB can hold, which is 4,294,967,295 (2 ³² - 1) bytes
Null	Represents a null value that can be assigned to the value of an OracleBlob instance

OracleBlob Static Methods

OracleBlob static methods are listed in [Table 13-12](#).

Table 13-12 OracleBlob Static Methods

Methods	Description
<code>Equals</code>	Inherited from <code>System.Object</code> (Overloaded)

OracleBlob Instance Properties

OracleBlob instance properties are listed in [Table 13-13](#).

Table 13-13 OracleBlob Instance Properties

Properties	Description
CanRead	Indicates whether or not the LOB stream can be read
CanSeek	Indicates whether or not forward and backward seek operations be performed
CanWrite	Indicates whether or not the LOB object supports writing
Connection	Indicates the <code>OracleConnection</code> that is used to retrieve and write BLOB data
IsEmpty	Indicates whether the BLOB is empty or not
IsInChunkWriteMode	Indicates whether or not the BLOB has been opened to defer index updates
IsNull	Indicates whether or not the current instance has a null value
IsTemporary	Indicates whether or not the current instance is bound to a temporary BLOB
Length	Indicates the size of the BLOB data
OptimumChunkSize	Indicates the optimal data buffer length (or multiples thereof) that read and write operations should use to improve performance
Position	Indicates the current read or write position in the LOB stream
Value	Returns the data, starting from the first byte in BLOB, as a byte array

OracleBlob Instance Methods

OracleBlob instance methods are listed in [Table 13-14](#).

Table 13-14 OracleBlob Instance Methods

Methods	Description
Append	Appends the supplied data to the current <code>OracleBlob</code> instance (Overloaded)
BeginChunkWrite	Opens the BLOB
<code>BeginRead</code>	Inherited from <code>System.IO.Stream</code>
<code>BeginWrite</code>	Inherited from <code>System.IO.Stream</code>
Clone	Creates a copy of an <code>OracleBlob</code> object

Table 13-14 (Cont.) OracleBlob Instance Methods

Methods	Description
Close	Closes the current stream and releases any resources associated with it
Compare	Compares data referenced by the current instance and that of the supplied object
CopyTo	Copies from the current OracleBlob instance to an OracleBlob object (Overloaded)
CopyToAsync	Returns a Task-based asynchronous version of OracleBlob.CopyTo() (Overloaded)
CreateObjRef	Inherited from System.MarshalByRefObject
Dispose	Releases resources allocated by this object
EndChunkWrite	Closes the BLOB referenced by the current OracleBlob instance
EndRead	Inherited from System.IO.Stream
EndWrite	Inherited from System.IO.Stream
Equals	Inherited from System.Object (Overloaded)
Erase	Erases data (Overloaded)
Flush	<i>Not supported</i>
GetHashCode	Inherited from System.Object
GetLifetimeService	Inherited from System.MarshalByRefObject
GetType	Inherited from System.Object
InitializedLifetimeService	Inherited from System.MarshalByRefObject
IsEqual	Compares the LOB data referenced by the two OracleBlobs
Read	Reads a specified amount of bytes from the ODP.NET LOB Type instance and populates the buffer
ReadAsync	Returns a Task-based asynchronous version of OracleBlob.Read(), which reads a specified number of bytes from the ODP.NET LOB instance and populates the <i>buffer</i> (Overloaded)
ReadByte	Inherited from System.IO.Stream
Search	Searches for a binary pattern in the current instance of an OracleBlob
Seek	Sets the position in the current LOB stream
SetLength	Trims or truncates the BLOB value to the specified length
ToString	Inherited from System.Object
Write	Writes the supplied buffer into the OracleBlob
WriteAsync	Returns a Task-based asynchronous version of OracleBlob.Write().
WriteByte	Inherited from System.IO.Stream

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Members](#)

OracleBlob Constructors

OracleBlob constructors are listed in [Table 13-10](#).

Overload List:

- [OracleBlob\(OracleConnection\)](#)

This constructor creates an instance of the OracleBlob class bound to a temporary BLOB with an OracleConnection object.

- [OracleBlob\(OracleConnection, bool\)](#)

This constructor creates an instance of the OracleBlob class bound to a temporary BLOB with an OracleConnection object and a boolean value for caching.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

OracleBlob(OracleConnection)

This constructor creates an instance of the OracleBlob class bound to a temporary BLOB with an OracleConnection object.

Declaration

```
// C#  
public OracleBlob(OracleConnection con);
```

Parameters

- *con*

The OracleConnection object.

Exceptions

InvalidOperationException - The OracleConnection is not opened.

Remarks

The connection must be opened explicitly by the application. OracleBlob does not open the connection implicitly.

The temporary BLOB utilizes the provided connection to store BLOB data. Caching is not turned on by this constructor.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

OracleBlob(OracleConnection, bool)

This constructor creates an instance of the `OracleBlob` class bound to a temporary BLOB with an `OracleConnection` object and a boolean value for caching.

Declaration

```
// C#  
public OracleBlob(OracleConnection con, bool bCaching);
```

Parameters

- *con*
The `OracleConnection` object.
- *bCaching*
A flag for enabling or disabling server-side caching.

Exceptions

`InvalidOperationException` - The `OracleConnection` is not opened.

Remarks

The connection must be opened explicitly by the application. `OracleBlob` does not open the connection implicitly.

The temporary BLOB uses the provided connection to store BLOB data. The *bCaching* input parameter determines whether or not server-side caching is used.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

OracleBlob Static Fields

`OracleBlob` static fields are listed in [Table 13-15](#).

Table 13-15 OracleBlob Static Fields

Field	Description
MaxSize	Holds the maximum number of bytes a BLOB can hold, which is 4,294,967,295 ($2^{32} - 1$) bytes
Null	Represents a null value that can be assigned to the value of an OracleBlob instance

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleBlob Class](#)
- [OracleBlob Members](#)

MaxSize

The MaxSize field holds the maximum number of bytes a BLOB can hold, which is 4,294,967,295 ($2^{32} - 1$) bytes.

Declaration

```
// C#  
public static readonly Int64 MaxSize = 4294967295;
```

Remarks

This field can be useful in code that checks whether or not the operation exceeds the maximum length allowed.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleBlob Class](#)
- [OracleBlob Members](#)

Null

This static field represents a null value that can be assigned to the value of an OracleBlob instance.

Declaration

```
// C#  
public static readonly OracleBlob Null;
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

OracleBlob Static Methods

OracleBlob static methods are listed in [Table 13-16](#).

Table 13-16 OracleBlob Static Methods

Methods	Description
Equals	Inherited from System.Object (Overloaded)

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

OracleBlob Instance Properties

OracleBlob instance properties are listed in [Table 13-17](#).

Table 13-17 OracleBlob Instance Properties

Properties	Description
CanRead	Indicates whether or not the LOB stream can be read
CanSeek	Indicates whether or not forward and backward seek operations be performed
CanWrite	Indicates whether or not the LOB object supports writing
Connection	Indicates the OracleConnection that is used to retrieve and write BLOB data
IsEmpty	Indicates whether the BLOB is empty or not
IsInChunkWriteMode	Indicates whether or not the BLOB has been opened to defer index updates
IsNull	Indicates whether or not the current instance has a null value
IsTemporary	Indicates whether or not the current instance is bound to a temporary BLOB
Length	Indicates the size of the BLOB data

Table 13-17 (Cont.) OracleBlob Instance Properties

Properties	Description
OptimumChunkSize	Indicates the optimal data buffer length (or multiples thereof) that read and write operations should use to improve performance
Position	Indicates the current read or write position in the LOB stream
Value	Returns the data, starting from the first byte in BLOB, as a byte array

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleBlob Class](#)
- [OracleBlob Members](#)

CanRead

Overrides Stream

This instance property indicates whether or not the LOB stream can be read.

Declaration

```
// C#  
public override bool CanRead{get;}
```

Property Value

If the LOB stream can be read, returns true; otherwise, returns false.

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleBlob Class](#)
- [OracleBlob Members](#)

CanSeek

Overrides Stream

This instance property indicates whether or not forward and backward seek operations can be performed.

Declaration

```
// C#  
public override bool CanSeek{get;}
```

Property Value

If forward and backward seek operations can be performed, returns true; otherwise, returns false.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleBlob Class](#)
- [OracleBlob Members](#)

CanWrite

Overrides Stream

This instance property indicates whether or not the LOB object supports writing.

Declaration

```
// C#  
public override bool CanWrite{get;}
```

Property Value

If the LOB stream can be written, returns true; otherwise, returns false.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleBlob Class](#)
- [OracleBlob Members](#)

Connection

This instance property indicates the OracleConnection that is used to retrieve and write BLOB data.

Declaration

```
// C#  
public OracleConnection Connection {get;}
```

Property Value

An object of OracleConnection.

Exceptions

ObjectDisposedException - The object is already disposed.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

IsEmpty

This instance property indicates whether the BLOB is empty or not.

Declaration

```
// C#  
public bool IsEmpty {get;}
```

Property Value

A bool that indicates whether or not the BLOB is empty.

Exceptions

ObjectDisposedException - The object is already disposed.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

IsInChunkWriteMode

This instance property indicates whether or not the BLOB has been opened to defer index updates.

Declaration

```
// C#  
public bool IsInChunkWriteMode{get;}
```

Property Value

If the BLOB has been opened, returns true; otherwise, returns false.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

IsNull

This property indicates whether or not the current instance has a null value.

Declaration

```
// C#  
public bool IsNull{get;}
```

Property Value

Returns true if the current instance has a null value; otherwise, returns false.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

IsTemporary

This instance property indicates whether or not the current instance is bound to a temporary BLOB.

Declaration

```
// C#  
public bool IsTemporary {get;}
```

Property Value

bool

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

Length

Overrides Stream

This instance property indicates the size of the BLOB data in bytes.

Declaration

```
// C#  
public override Int64 Length {get;}
```

Property Value

A number indicating the size of the BLOB data in bytes.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

OptimumChunkSize

This instance property indicates the optimal data buffer length (or multiples thereof) that read and write operations should use to improve performance.

Declaration

```
// C#  
public int OptimumChunkSize{get;}
```

Property Value

A number representing the minimum bytes to retrieve or send.

Exceptions

ObjectDisposedException - The object is already disposed.

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleBlob Class](#)
- [OracleBlob Members](#)

Position

Overrides Stream

This instance property indicates the current read or write position in the LOB stream.

Declaration

```
// C#  
public override Int64 Position{get; set;}
```

Property Value

An Int64 that indicates the read or write position.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

ArgumentOutOfRangeException - The Position is less than 0.

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleBlob Class](#)
- [OracleBlob Members](#)

Value

This instance property returns the data, starting from the first byte in the BLOB, as a byte array.

Declaration

```
// C#  
public Byte[] Value{get;}
```

Property Value

A byte array.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - The Value is less than 0.

Remarks

The value of `Position` is not used or changed by using this property. 2 GB is the maximum byte array length that can be returned by this property.

For very large values, .NET may not be able to allocate enough contiguous memory to store all the data. Typically, you will see a `System.OverflowException` or `System.OutOfMemoryException` error then. To retrieve the large value, you can retrieve the data in chunks by calling `GetChars()` or `GetBytes()` method on the `OracleDataReader`. You can also call `Read()` method on the class.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

OracleBlob Instance Methods

OracleBlob instance methods are listed in [Table 13-18](#).

Table 13-18 OracleBlob Instance Methods

Methods	Description
Append	Appends the supplied data to the current <code>OracleBlob</code> instance (Overloaded)
BeginChunkWrite	Opens the BLOB
<code>BeginRead</code>	Inherited from <code>System.IO.Stream</code>
<code>BeginWrite</code>	Inherited from <code>System.IO.Stream</code>
Clone	Creates a copy of an <code>OracleBlob</code> object
Close	Closes the current stream and releases any resources associated with it
Compare	Compares data referenced by the current instance and that of the supplied object
CopyTo	Copies from the current <code>OracleBlob</code> instance to an <code>OracleBlob</code> object (Overloaded)
CopyToAsync	Returns a Task-based asynchronous version of <code>OracleBlob.CopyTo()</code> (Overloaded)
<code>CreateObjRef</code>	Inherited from <code>System.MarshalByRefObject</code>

Table 13-18 (Cont.) OracleBlob Instance Methods

Methods	Description
Dispose	Releases resources allocated by this object
EndChunkWrite	Closes the BLOB referenced by the current OracleBlob instance
EndRead	Inherited from System.IO.Stream
EndWrite	Inherited from System.IO.Stream
Equals	Inherited from System.Object (Overloaded)
Erase	Erases data (Overloaded)
Flush	<i>Not supported</i>
GetHashCode	Inherited from System.Object
GetLifetimeService	Inherited from System.MarshalByRefObject
GetType	Inherited from System.Object
InitializedLifetimeService	Inherited from System.MarshalByRefObject
IsEqual	Compares the LOB data referenced by the two OracleBlobs
Read	Reads a specified amount of bytes from the ODP.NET LOB Type instance and populates the buffer
ReadAsync	Returns a Task-based asynchronous version of OracleBlob.Read(), which reads a specified number of bytes from the ODP.NET LOB instance and populates the <i>buffer</i> (Overloaded)
ReadByte	Inherited from System.IO.Stream
Search	Searches for a binary pattern in the current instance of an OracleBlob
Seek	Sets the position in the current LOB stream
SetLength	Trims or truncates the BLOB value to the specified length
ToString	Inherited from System.Object
Write	Writes the supplied buffer into the OracleBlob
WriteAsync	Returns a Task-based asynchronous version of OracleBlob.Write().
WriteByte	Inherited from System.IO.Stream

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleBlob Class](#)
- [OracleBlob Members](#)

Append

Append appends the supplied data to the end of the current OracleBlob instance.

Overload List:

- [Append\(OracleBlob\)](#)
This instance method appends the BLOB data referenced by the provided `OracleBlob` object to the current `OracleBlob` instance.
- [Append\(byte\[\], int, int\)](#)
This instance method appends data from the supplied byte array buffer to the end of the current `OracleBlob` instance.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

Append(OracleBlob)

This instance method appends the BLOB data referenced by the provided `OracleBlob` object to the current `OracleBlob` instance.

Declaration

```
// C#  
public void Append(OracleBlob obj);
```

Parameters

- *obj*
An object of `OracleBlob`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The parameter has a different connection than the object, `OracleConnection` is not opened, or `OracleConnection` has been reopened.

Remarks

No character set conversions are made.

The provided object and the current instance must be using the same connection; that is, the same `OracleConnection` object.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

Append(byte[], int, int)

This instance method appends data from the supplied byte array buffer to the end of the current OracleBlob instance.

Declaration

```
// C#  
public void Append(byte[] buffer, int offset, int count);
```

Parameters

- *buffer*
An array of bytes.
- *offset*
The zero-based byte offset in the buffer from which data is read.
- *count*
The number of bytes to be appended.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Client;  
using Oracle.DataAccess.Types;  
  
class AppendSample  
{  
    static void Main()  
    {  
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";  
        OracleConnection con = new OracleConnection(constr);  
        con.Open();  
  
        OracleBlob blob = new OracleBlob(con);  
  
        // Append 2 bytes {4, 5} to the OracleBlob  
        byte[] buffer = new byte[3] {4, 5, 6};  
        blob.Append(buffer, 0, 2);  
    }  
}
```

```
byte[] appendBuffer = blob.Value;

// Prints "appendBuffer = 45"
Console.Write("appendBuffer = ");
for(int index = 0; index < appendBuffer.Length; index++)
{
    Console.Write(appendBuffer[index]);
}
Console.WriteLine();

blob.Close();
blob.Dispose();

con.Close();
con.Dispose();
}
}
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

BeginChunkWrite

This instance method opens the BLOB.

Declaration

```
// C#
public void BeginChunkWrite();
```

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

BeginChunkWrite does not need to be called before manipulating the BLOB data. This is provided for performance reasons.

After this method is called, write operations do not cause the domain or function-based index on the column to be updated. Index updates occur only once after EndChunkWrite is called.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

Clone

This instance method creates a copy of an OracleBlob object.

Declaration

```
// C#  
public object Clone();
```

Return Value

An OracleBlob object.

Implements

ICloneable

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

The cloned object has the same property values as that of the object being cloned.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Client;  
using Oracle.DataAccess.Types;  
  
class CloneSample  
{  
    static void Main()  
    {  
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";  
        OracleConnection con = new OracleConnection(constr);  
        con.Open();  
  
        OracleBlob blob1 = new OracleBlob(con);  
  
        // Prints "blob1.Position = 0"  
        Console.WriteLine("blob1.Position = " + blob1.Position);  
  
        // Set the Position before calling Clone()  
        blob1.Position = 1;
```

```
// Clone the OracleBlob
OracleBlob blob2 = (OracleBlob)blob1.Clone();

// Prints "blob2.Position = 1"
Console.WriteLine("blob2.Position = " + blob2.Position);

blob1.Close();
blob1.Dispose();

blob2.Close();
blob2.Dispose();

con.Close();
con.Dispose();
}
}
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

Close

Overrides Stream

This instance method closes the current stream and releases any resources associated with it.

Declaration

```
// C#
public override void Close();
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

Compare

This instance method compares data referenced by the current instance and that of the supplied object.

Declaration

```
// C#
public int Compare(Int64 src_offset, OracleBlob obj, Int64 dst_offset,
    Int64 amount);
```

Parameters

- *src_offset*
The comparison starting point (in bytes) for the current instance.
- *obj*
The provided OracleBlob object.
- *dst_offset*
The comparison starting point (in bytes) for the provided OracleBlob.
- *amount*
The number of bytes to compare.

Return Value

Returns a value that is:

- Less than zero: if the data referenced by the current instance is less than that of the supplied instance
- Zero: if both objects reference the same data
- Greater than zero: if the data referenced by the current instance is greater than that of the supplied instance

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The parameter has a different connection than the object, OracleConnection is not opened, or OracleConnection has been reopened.

ArgumentOutOfRangeException - The *src_offset*, the *dst_offset*, or the *amount* parameter is less than 0.

Remarks

The provided object and the current instance must be using the same connection, that is, the same OracleConnection object.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

CopyTo

CopyTo copies data from the current instance to the provided OracleBlob object.

Overload List:

- [CopyTo\(OracleBlob\)](#)

This instance method copies data from the current instance to the provided `OracleBlob` object.

- [CopyTo\(OracleBlob, Int64\)](#)

This instance method copies data from the current `OracleBlob` instance to the provided `OracleBlob` object with the specified destination offset.

- [CopyTo\(Int64, OracleBlob, Int64, Int64\)](#)

This instance method copies data from the current `OracleBlob` instance to the provided `OracleBlob` object with the specified source offset, destination offset, and character amounts.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

CopyTo(OracleBlob)

This instance method copies data from the current instance to the provided `OracleBlob` object.

Declaration

```
// C#  
public Int64 CopyTo(OracleBlob obj);
```

Parameters

- *obj*
The `OracleBlob` object to which the data is copied.

Return Value

The return value is the amount copied.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - This exception is thrown if any of the following conditions exist:

- The `OracleConnection` is not open or has been closed during the lifetime of the object.
- The LOB object parameter has a different connection than the object.

Remarks

The provided object and the current instance must be using the same connection; that is, the same `OracleConnection` object.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

CopyTo(OracleBlob, Int64)

This instance method copies data from the current `OracleBlob` instance to the provided `OracleBlob` object with the specified destination offset.

Declaration

```
// C#  
public Int64 CopyTo(OracleBlob obj, Int64 dst_offset);
```

Parameters

- *obj*
The `OracleBlob` object to which the data is copied.
- *dst_offset*
The offset (in bytes) at which the `OracleBlob` object is copied.

Return Value

The return value is the amount copied.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentOutOfRangeException` - The *dst_offset* is less than 0.

`InvalidOperationException` - This exception is thrown if any of the following conditions exist:

- The `OracleConnection` is not open or has been closed during the lifetime of the object.
- The LOB object parameter has a different connection than the object.

Remarks

If the *dst_offset* is beyond the end of the `OracleBlob` data, spaces are written into the `OracleBlob` until the *dst_offset* is met.

The offsets are 0-based. No character conversion is performed by this operation.

The provided object and the current instance must be using the same connection; that is, the same `OracleConnection` object.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

CopyTo(Int64, OracleBlob, Int64, Int64)

This instance method copies data from the current `OracleBlob` instance to the provided `OracleBlob` object with the specified source offset, destination offset, and character amounts.

Declaration

```
// C#  
public Int64 CopyTo(Int64 src_offset, OracleBlob obj, Int64 dst_offset,  
    Int64 amount);
```

Parameters

- *src_offset*
The offset (in bytes) in the current instance, from which the data is read.
- *obj*
The `OracleBlob` object to which the data is copied.
- *dst_offset*
The offset (in bytes) at which the `OracleBlob` object is copied.
- *amount*
The amount of data to be copied.

Return Value

The return value is the amount copied.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The parameter has a different connection than the object, `OracleConnection` is not opened, or `OracleConnection` has been reopened.

`ArgumentOutOfRangeException` - The *src_offset*, the *dst_offset*, or the *amount* parameter is less than 0.

Remarks

If the *dst_offset* is beyond the end of the `OracleBlob` data, spaces are written into the `OracleBlob` until the *dst_offset* is met.

The offsets are 0-based. No character conversion is performed by this operation.

The provided object and the current instance must be using the same connection; that is, the same `OracleConnection` object.

Example

```
// C#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class CopyToSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleBlob blob1 = new OracleBlob(con);
        OracleBlob blob2 = new OracleBlob(con);

        // Write 4 bytes, starting at buffer offset 0
        byte[] buffer = new byte[4] {1, 2, 3, 4};
        blob1.Write(buffer, 0, 4);

        // Copy 2 bytes from byte 0 of blob1 to byte 1 of blob2
        blob1.CopyTo(0, blob2, 1, 2);

        byte[] copyBuffer = blob2.Value;

        //Prints "Value = 012"
        Console.Write("Value = ");
        for(int index = 0; index < copyBuffer.Length; index++)
        {
            Console.Write(copyBuffer[index]);
        }
        Console.WriteLine();

        blob1.Close();
        blob1.Dispose();

        blob2.Close();
        blob2.Dispose();

        con.Close();
        con.Dispose();
    }
}
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

CopyToAsync

CopyToAsync returns a Task-based asynchronous version of `OracleBlob.CopyTo()`.

Overload List:

- [CopyToAsync\(Int64, OracleBlob, Int64, Int64\)](#)
This instance method returns a Task-based asynchronous version of `OracleBlob.CopyTo()`.
- [CopyToAsync\(Int64, OracleBlob, Int64, Int64, CancellationToken\)](#)
This instance method returns a Task-based asynchronous version of `OracleBlob.CopyTo()`.
- [CopyToAsync\(OracleBlob, Int64\)](#)
This instance method returns a Task-based asynchronous version of `OracleBlob.CopyTo()`.
- [CopyToAsync\(OracleBlob, Int64, CancellationToken\)](#)
This instance method returns a Task-based asynchronous version of `OracleBlob.CopyTo()`.
- [CopyToAsync\(OracleBlob\)](#)
This instance method returns a Task-based asynchronous version of `OracleBlob.CopyTo()`.
- [CopyToAsync\(OracleBlob, CancellationToken\)](#)
This instance method returns a Task-based asynchronous version of `OracleBlob.CopyTo()`.

Example (includes all Overloads)

```
using Oracle.ManagedDataAccess.Client;
using Oracle.ManagedDataAccess.Types;
using System;
using System.Threading;
using System.Threading.Tasks;

namespace AsyncApp
{
    class AsyncDemo
    {
        static async Task Main(string[] args)
        {
            string connectionString = "User Id=HR; Password=<PASSWORD>; Data Source=oracle;";

            OracleConnection oc = new OracleConnection(connectionString);
            await oc.OpenAsync(CancellationToken.None);

            OracleCommand cmd = oc.CreateCommand();
            cmd.CommandText = " select blob_column from tab1";

            OracleBlob blob1 = new OracleBlob(oc);
            OracleBlob blob2 = new OracleBlob(oc);
            OracleBlob blob3 = new OracleBlob(oc);
            OracleBlob blob4 = new OracleBlob(oc);
            OracleBlob blob5 = new OracleBlob(oc);
            OracleBlob blob6 = new OracleBlob(oc);

            OracleDataReader reader = await cmd.ExecuteReaderAsync();
```



```

        await reader.ReadAsync(CancellationTokens.None) ;
        using (OracleBlob blob = reader.GetOracleBlob(0))
        {
            //asynchronously copy blob data
            Int64 bytesCopied1 = await blob.CopyToAsync(0, blob1, 0, blob.Length);
            Console.WriteLine("bytes copied to blob1 = " + bytesCopied1);

            //asynchronously copy blob data
            Int64 bytesCopied2 = await blob.CopyToAsync(0, blob2, 0, blob.Length,
CancellationTokens.None);
            Console.WriteLine("bytes copied to blob2 = " + bytesCopied2);

            //asynchronously copy blob data
            Int64 bytesCopied3 = await blob.CopyToAsync(blob3, 0);
            Console.WriteLine("bytes copied to blob3 = " + bytesCopied3);

            //asynchronously copy blob data
            Int64 bytesCopied4 = await blob.CopyToAsync(blob4, 0, CancellationTokens.None);
            Console.WriteLine("bytes copied to blob4 = " + bytesCopied4);

            //asynchronously copy blob data
            Int64 bytesCopied5 = await blob.CopyToAsync(blob5);
            Console.WriteLine("bytes copied to blob5 = " + bytesCopied5);

            //asynchronously copy blob data
            Int64 bytesCopied6 = await blob.CopyToAsync(blob6, CancellationTokens.None);
            Console.WriteLine("bytes copied to blob6 = " + bytesCopied6);
        }
    }
}
}
}

```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

CopyToAsync(Int64, OracleBlob, Int64, Int64)

The CopyToAsync method returns a Task-based asynchronous version of OracleBlob.CopyTo().

Declaration

```
// C#
public Task<Int64> CopyToAsync(Int64 src_offset, OracleBlob obj, Int64 dst_offset, Int64 amount);
```

Parameters

- *src_offset*

The offset (in bytes) in the current instance, from which the data is read.

- *obj*

The OracleBlob object to which the data is copied.

- *dst_offset*

The offset (in bytes) at which the OracleBlob object is copied.

- *amount*

The amount of data to be copied.

Return Value

Task<Int64> object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `ArgumentOutOfRangeException` - The *src_offset*, the *dst_offset*, or the *amount* parameter is less than 0.
- `InvalidOperationException` - This exception is thrown if any of the following conditions exist:
 - The `OracleConnection` is not open or has been closed during the lifetime of the object.
 - The LOB object parameter has a different connection than the object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

CopyToAsync(Int64, OracleBlob, Int64, Int64, CancellationToken)

The `CopyToAsync` method returns a Task-based asynchronous version of `OracleBlob.CopyTo()`.

Declaration

```
// C#
public Task<Int64> CopyToAsync(Int64 src_offset, OracleBlob obj, Int64 dst_offset, Int64 amount, CancellationToken cancellationToken);
```

Parameters

- *src_offset*

The offset (in bytes) in the current instance, from which the data is read.

- *obj*

The OracleBlob object to which the data is copied.

- *dst_offset*

The offset (in bytes) at which the OracleBlob object is copied.

- *amount*
The amount of data to be copied.
- *cancellationToken*
The input cancellation token which can be used by the application to cancel the task

Return Value

Task<Int64> object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `ArgumentOutOfRangeException` - The *src_offset*, the *dst_offset*, or the *amount* parameter is less than 0.
- `InvalidOperationException` - This exception is thrown if any of the following conditions exist:
 - The `OracleConnection` is not open or has been closed during the lifetime of the object.
 - The LOB object parameter has a different connection than the object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

CopyToAsync(OracleBlob, Int64)

The `CopyToAsync` method returns a Task-based asynchronous version of `OracleBlob.CopyTo()`.

Declaration

```
// C#  
public async Task<Int64> CopyToAsync(OracleBlob obj, Int64 dst_offset);
```

Parameters

- *obj*
The `OracleBlob` object to which the data is copied.
- *dst_offset*
The offset (in bytes) at which the `OracleBlob` object is copied.

Return Value

Task<Int64> object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `ArgumentOutOfRangeException` - The *dst_offset* parameter is less than 0.
- `InvalidOperationException` - This exception is thrown if any of the following conditions exist:
 - The `OracleConnection` is not open or has been closed during the lifetime of the object.
 - The LOB object parameter has a different connection than the object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

CopyToAsync(OracleBlob, Int64, CancellationToken)

The `CopyToAsync` method returns a Task-based asynchronous version of `OracleBlob.CopyTo()`.

Declaration

```
// C#
public async Task<Int64> CopyToAsync(OracleBlob obj, Int64 dst_offset, CancellationToken
cancellation_token);
```

Parameters

- *obj*
The `OracleBlob` object to which the data is copied.
- *dst_offset*
The offset (in bytes) at which the `OracleBlob` object is copied.
- *cancellation_token*
The input cancellation token which can be used by the application to cancel the task

Return Value

`Task<Int64>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `ArgumentOutOfRangeException` - The *dst_offset* is less than 0.
- `InvalidOperationException` - This exception is thrown if any of the following conditions exist:
 - The `OracleConnection` is not open or has been closed during the lifetime of the object.

- The LOB object parameter has a different connection than the object.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

CopyToAsync(OracleBlob)

The CopyToAsync method returns a Task-based asynchronous version of `OracleBlob.CopyTo()`.

Declaration

```
// C#  
public async Task<Int64> CopyToAsync(OracleBlob obj);
```

Parameters

- *obj*
The `OracleBlob` object to which the data is copied.

Return Value

`Task<Int64>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `InvalidOperationException` - This exception is thrown if any of the following conditions exist:
 - The `OracleConnection` is not open or has been closed during the lifetime of the object.
 - The LOB object parameter has a different connection than the object.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

CopyToAsync(OracleBlob, CancellationToken)

The CopyToAsync method returns a Task-based asynchronous version of `OracleBlob.CopyTo()`.

Declaration

```
// C#  
public async Task<Int64> CopyToAsync(OracleBlob obj, CancellationToken  
cancellation_token);
```

Parameters

- *obj*
The OracleBlob object to which the data is copied.
- *cancellation_token*
The input cancellation token which can be used by the application to cancel the task

Return Value

Task<Int64> object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `InvalidOperationException` - This exception is thrown if any of the following conditions exist:
 - The `OracleConnection` is not open or has been closed during the lifetime of the object.
 - The LOB object parameter has a different connection than the object.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

Dispose

This instance method releases resources allocated by this object.

Declaration

```
// C#  
public void Dispose();
```

Implements

`IDisposable`

Remarks

Once `Dispose()` is called, the object of `OracleBlob` is in an uninitialized state.

Although some properties can still be accessed, their values may not be accountable. Since resources are freed, method calls may lead to exceptions. The object cannot be reused after being disposed.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

EndChunkWrite

This instance method closes the BLOB referenced by the current OracleBlob instance.

Declaration

```
// C#  
public void EndChunkWrite();
```

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

Index updates occur immediately if there is write operation(s) deferred by the BeginChunkWrite method.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

Erase

Erase erases a portion or all data.

Overload List:

- [Erase\(\)](#)

This instance method erases all data.

- [Erase\(Int64, Int64\)](#)

This instance method erases a specified portion of data.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

Erase()

This instance method erases all data.

Declaration

```
// C#  
public Int64 Erase();
```

Return Value

The number of bytes erased.

Remarks

Erase() replaces all data with zero-byte fillers.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

Erase(Int64, Int64)

This instance method erases a specified portion of data.

Declaration

```
// C#  
public Int64 Erase(Int64 offset, Int64 amount);
```

Parameters

- *offset*
The offset from which to erase.
- *amount*
The quantity (in bytes) to erase.

Return Value

The number of bytes erased.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

ArgumentOutOfRangeException - The *offset* or *amount* parameter is less than 0.

Remarks

Replaces the specified *amount* of data with zero-byte fillers.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

Flush

This method is not supported.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

IsEqual

This instance method compares the LOB data referenced by the two OracleBlobs.

Declaration

```
// C#  
public bool IsEqual(OracleBlob obj);
```

Parameters

- *obj*
An OracleBlob object.

Return Value

If the current OracleBlob and the provided OracleBlob refer to the same LOB, returns true. Returns false otherwise.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

Note that this method can return true even if the two OracleBlob objects return false for == or Equals() because two different OracleBlob instances can refer to the same LOB.

The provided object and the current instance must be using the same connection, that is, the same OracleConnection object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

Read

Overrides Stream

This instance method reads a specified amount of bytes from the ODP.NET LOB instance and populates the buffer.

Declaration

```
// C#  
public override int Read(byte[] buffer, int offset, int count);
```

Parameters

- *buffer*
The byte array buffer to be populated.
- *offset*
The starting offset (in bytes) at which the buffer is populated.
- *count*
The amount of bytes to read.

Return Value

The return value indicates the number of bytes read from the LOB.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

ArgumentOutOfRangeException - This exception is thrown if any of the following conditions exist:

- The *offset* or the *count* parameter is less than 0.
- The *offset* is greater than or equal to the *buffer.Length*.
- The *offset* and the *count* together are greater than the *buffer.Length*.

Remarks

The LOB data is read starting from the position specified by the *Position* property.

Example

```
// C#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class ReadSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleBlob blob = new OracleBlob(con);

        // Write 3 bytes, starting at buffer offset 1
        byte[] writeBuffer = new byte[4] {1, 2, 3, 4};
        blob.Write(writeBuffer, 1, 3);

        // Reset the Position for Read
        blob.Position = 1;

        // Read 2 bytes into buffer starting at buffer offset 1
        byte[] readBuffer = new byte[4];
        int bytesRead = blob.Read(readBuffer, 1, 2);

        // Prints "bytesRead = 2"
        Console.WriteLine("bytesRead = " + bytesRead);

        // Prints "readBuffer = 0340"
        Console.Write("readBuffer = ");
        for(int index = 0; index < readBuffer.Length; index++)
        {
            Console.Write(readBuffer[index]);
        }
        Console.WriteLine();

        blob.Close();
        blob.Dispose();

        con.Close();
        con.Dispose();
    }
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

ReadAsync

ReadAsync returns a Task-based asynchronous version of `OracleBlob.Read()`, which reads a specified number of bytes from the ODP.NET LOB instance and populates the *buffer*.

Overload List:

- [ReadAsync\(byte\[\], int, int\)](#)
This instance method returns a Task-based asynchronous version of `OracleBlob.Read()`.
- [ReadAsync\(byte\[\], int, int, CancellationToken\)](#)
This instance method returns a Task-based asynchronous version of `OracleBlob.Read()`.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

ReadAsync(byte[], int, int)

The ReadAsync method returns a Task-based asynchronous version of `OracleBlob.Read()`, which reads a specified number of bytes from the ODP.NET LOB instance and populates the buffer.

Declaration

```
// C#  
public Task<int> ReadAsync(byte[] buffer, int offset, int count);
```

Parameters

- *buffer*
The byte array buffer to be populated.
- *offset*
The starting offset (in bytes) at which the buffer is populated.
- *count*
The number of bytes to read.

Return Value

`Task<Int>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object
- `ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:
 - The offset or the count parameter is less than 0.
 - The offset is greater than or equal to the `buffer.Length`.
 - The offset and the count together are greater than the `buffer.Length`.

Remark

This will call into the `ReadAsync` implementation with argument `cancellationToken` passed as `CancellationToken.None`.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

ReadAsync(byte[], int, int, CancellationToken)

The `ReadAsync` method returns a Task-based asynchronous version of `OracleBlob.Read()`, which reads a specified number of bytes from the ODP.NET LOB instance and populates the buffer.

Declaration

```
// C#  
public override Task<int> ReadAsync(byte[] buffer, int offset, int count,  
CancellationToken cancellationToken);
```

Parameters

- *buffer*
The byte array buffer to be populated.
- *offset*
The starting offset (in bytes) at which the buffer is populated.
- *count*
The number of bytes to read.

- *cancellationToken*

The input cancellation token which can be used by the application to cancel the task.

Return Value

`Task<Int>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object
- `ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:
 - The offset or the count parameter is less than 0.
 - The offset is greater than or equal to the `buffer.Length`.
 - The offset and the count together are greater than the `buffer.Length`.

Example

```
using Oracle.ManagedDataAccess.Client;
using Oracle.ManagedDataAccess.Types;
using System;
using System.Threading;
using System.Threading.Tasks;

namespace AsyncApp
{
    class AsyncDemo
    {
        static async Task Main(string[] args)
        {
            string connectionString = "User Id=HR; Password=<PASSWORD>; Data Source=oracle;";

            OracleConnection oc = new OracleConnection(connectionString);
            await oc.OpenAsync(CancellationToken.None);

            OracleCommand cmd = oc.CreateCommand();
            cmd.CommandText = " select blob_column from tab1";

            OracleDataReader reader = await cmd.ExecuteReaderAsync();

            while (await reader.ReadAsync(CancellationToken.None))
            {
                using (OracleBlob blob = reader.GetOracleBlob(0))
                {
                    byte[] buffer = new byte[100];
                    //asynchronously read blob data
                    Task<int> task = blob.ReadAsync(buffer, 0, 64, CancellationToken.None);

                    //other operations
                    Console.WriteLine("Hello World");

                    //await for asynchronous ReadAsync
                    int bytesRead = await task;
                }
            }
        }
    }
}
```

```
        Console.WriteLine("Bytes Read: " + bytesRead);  
    }  
}  
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

Search

This instance method searches for a binary pattern in the current instance of an `OracleBlob`.

Declaration

```
// C#  
public Int64 Search(byte[] val, int64 offset, int64 nth);
```

Parameters

- *val*
The binary pattern being searched for.
- *offset*
The 0-based offset (in bytes) starting from which the `OracleBlob` is searched.
- *nth*
The specific occurrence (1-based) of the match for which the absolute offset (in bytes) is returned.

Return Value

Returns the absolute *offset* of the start of the matched pattern (in bytes) for the *nth* occurrence of the match. Otherwise, 0 is returned.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:

- The *offset* is less than 0.
- The *nth* is less than or equal to 0.
- The *val.Length* is greater than 16383.
- The *nth* is greater than or equal to `OracleBlob.MaxSize`.

- The *offset* is greater than or equal to `OracleBlob.MaxValue`.

Remarks

The limit of the search pattern is 16383 bytes.

Example

```
// C#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class SearchSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleBlob blob = new OracleBlob(con);

        // Write 7 bytes, starting at buffer offset 0
        byte[] buffer = new byte[7] {1, 2, 3, 4, 1, 2, 3};
        blob.Write(buffer, 0, 7);

        // Search for the 2nd occurrence of a byte pattern '23'
        // starting at offset 1 in the OracleBlob
        byte[] pattern = new byte[2] {2, 3};
        long posFound = blob.Search(pattern, 1, 2);

        // Prints "posFound = 6"
        Console.WriteLine("posFound = " + posFound);

        blob.Close();
        blob.Dispose();

        con.Close();
        con.Dispose();
    }
}
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

Seek

Overrides `Stream`

This instance method sets the position on the current LOB stream.

Declaration

```
// C#  
public override Int64 Seek(Int64 offset, SeekOrigin origin);
```

Parameters

- *offset*
A byte offset relative to origin.
- *origin*
A value of type `System.IO.SeekOrigin` indicating the reference point used to obtain the new position.

Return Value

Returns `Int64` for the position.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

If *offset* is negative, the new position precedes the position specified by *origin* by the number of bytes specified by *offset*.

If *offset* is zero, the new position is the position specified by *origin*.

If *offset* is positive, the new position follows the position specified by *origin* by the number of bytes specified by *offset*.

`SeekOrigin.Begin` specifies the beginning of a stream.

`SeekOrigin.Current` specifies the current position within a stream.

`SeekOrigin.End` specifies the end of a stream.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

SetLength

Overrides `Stream`

This instance method trims or truncates the BLOB value to the specified length (in bytes).

Declaration

```
// C#  
public override void SetLength(Int64 newlen);
```

Parameters

- *newlen*

The desired length of the current stream in bytes.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

ArgumentOutOfRangeException - The *newlen* parameter is less than 0.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

Write

Overrides Stream

This instance method writes the supplied buffer into the OracleBlob.

Declaration

```
// C#  
public override void Write(byte[ ] buffer, int offset, int count);
```

Parameters

- *buffer*

The byte array buffer that provides the data.

- *offset*

The 0-based offset (in bytes) from which the buffer is read.

- *count*

The amount of data (in bytes) that is to be written into the OracleBlob.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

ArgumentOutOfRangeException - This exception is thrown if any of the following conditions exist:

- The *offset* or the *count* is less than 0.
- The *offset* is greater than or equal to the *buffer.Length*.
- The *offset* and the *count* together are greater than *buffer.Length*.

Remarks

Destination *offset* in the *OracleBlob* can be specified by the *Position* property.

Example

```
// C#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class WriteSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleBlob blob = new OracleBlob(con);

        // Set the Position for the Write
        blob.Position = 0;

        // Begin ChunkWrite to improve performance
        // Index updates occur only once after EndChunkWrite
        blob.BeginChunkWrite();

        // Write to the OracleBlob in 5 chunks of 2 bytes each
        byte[] b = new byte[2] {1, 2};
        for(int index = 0; index < 5; index++)
        {
            blob.Write(b, 0, b.Length);
        }
        blob.EndChunkWrite();

        byte[] chunkBuffer = blob.Value;

        // Prints "chunkBuffer = 1212121212"
        Console.Write("chunkBuffer = ");
        for(int index = 0; index < chunkBuffer.Length; index++)
        {
            Console.Write(chunkBuffer[index]);
        }
        Console.WriteLine();

        blob.Close();
        blob.Dispose();

        con.Close();
        con.Dispose();
    }
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

WriteAsync

WriteAsync returns a Task-based asynchronous version of `OracleBlob.Write()`.

Overload List:

- [WriteAsync\(byte\[\], int, int\)](#)
This instance method returns a Task-based asynchronous version of `OracleBlob.Write()`.
- [WriteAsync\(byte\[\], int, int, CancellationToken\)](#)
This instance method returns a Task-based asynchronous version of `OracleBlob.Write()`.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

WriteAsync(byte[], int, int)

The WriteAsync method returns a Task-based asynchronous version of `OracleBlob.Write()`.

Declaration

```
// C#  
public Task WriteAsync(byte[] buffer, int offset, int count);
```

Parameters

- *buffer*
The byte array buffer that provides the data.
- *offset*
The 0-based offset (in bytes) from which the buffer is read.
- *count*
The amount of data (in bytes) that is to be written into the `OracleBlob`.

Return Value

Task object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object
- `ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:
 - The offset or the count parameter is less than 0.
 - The offset is greater than or equal to the `buffer.Length`.
 - The offset and the count together are greater than the `buffer.Length`.

Remark

This will call into the `WriteAsync` implementation with argument `cancellationToken` passed as `CancellationToken.None`.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

WriteAsync(byte[], int, int, CancellationToken)

The `WriteAsync` method returns a Task-based asynchronous version of `OracleBlob.Write()`.

Declaration

```
// C#  
public override Task WriteAsync(byte[] buffer, int offset, int count, CancellationToken  
cancellationToken);
```

Parameters

- *buffer*
The byte array buffer that provides the data.
- *offset*
The 0-based offset (in bytes) from which the buffer is read.
- *count*
The amount of data (in bytes) that is to be written into the `OracleBlob`.
- *cancellationToken*
The input cancellation token which can be used by the application to cancel the task.

Return Value

Task object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object
- `ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:
 - The offset or the count parameter is less than 0.
 - The offset is greater than or equal to the `buffer.Length`.
 - The offset and the count together are greater than the `buffer.Length`.

Example

```
using Oracle.ManagedDataAccess.Client;
using Oracle.ManagedDataAccess.Types;
using System;
using System.Threading;
using System.Threading.Tasks;

namespace AsyncApp
{
    class AsyncDemo
    {
        static async Task Main(string[] args)
        {
            string connectionString = "User Id=HR; Password=<PASSWORD>; Data Source=oracle;";

            OracleConnection oc = new OracleConnection(connectionString);
            await oc.OpenAsync(CancellationToken.None);

            OracleBlob blob = new OracleBlob(oc);

            byte[] writeBuffer = new byte[4] { 1, 2, 3, 4 };

            //Write data to Binary Large Object, asynchronously
            Task task = blob.WriteAsync(writeBuffer, 0, 4, CancellationToken.None);

            //other operations
            Console.WriteLine("Hello World");

            //await for asynchronous WriteAsync
            await task;
            Console.WriteLine("blob.Length = " + blob.Length);
        }
    }
}
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBlob Class](#)
- [OracleBlob Members](#)

OracleClob Class

An OracleClob is an object that has a reference to CLOB data. It provides methods for performing operations on CLOBs.

Note

The OracleClob object uses the client side character set when retrieving or writing CLOB data using a .NET Framework byte array.

Class Inheritance

System.Object
 System.MarshalByRefObject
 System.IO.Stream
 Oracle.DataAccess.Types.OracleClob

Declaration

```
// C#
public sealed class OracleClob : Stream, ICloneable, INullable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class OracleClobSample
{
    static void Main()
```

```
{
    string constr = "User Id=scott;Password=tiger;Data Source=oracle";
    OracleConnection con = new OracleConnection(constr);
    con.Open();

    OracleClob clob = new OracleClob(con);

    // Write 4 chars from writeBuffer, starting at buffer offset 0
    char[] writeBuffer = new char[4] {'a', 'b', 'c', 'd'};
    clob.Write(writeBuffer, 0, 4);

    // Append first 2 chars from writeBuffer {'a', 'b'} to the oracleClob
    clob.Append(writeBuffer, 0, 2);

    // Prints "clob.Length = 12"
    Console.WriteLine("clob.Length = " + clob.Length);

    // Reset the Position for the Read
    clob.Position = 0;

    // Read 6 chars into readBuffer, starting at buffer offset 0
    char[] readBuffer = new char[6];
    int charsRead = clob.Read(readBuffer, 0, 6);

    // Prints "charsRead = 6"
    Console.WriteLine("charsRead = " + charsRead);

    // Prints "readBuffer = abcdab"
    Console.Write("readBuffer = ");
    for(int index = 0; index < readBuffer.Length; index++)
    {
        Console.Write(readBuffer[index]);
    }
    Console.WriteLine();

    // Search for the 2nd occurrence of a char pattern 'ab'
    // starting from char offset 0 in the OracleClob
    char[] pattern = new char[2] {'a', 'b'};
    long posFound = clob.Search(pattern, 0, 2);

    // Prints "posFound = 5"
    Console.WriteLine("posFound = " + posFound);

    // Erase 4 chars of data starting at char offset 1
    // Sets chars to ''
    clob.Erase(1, 4);

    //Prints "clob.Value = a b"
    Console.WriteLine("clob.Value = " + clob.Value);

    clob.Close();
    clob.Dispose();

    con.Close();
    con.Dispose();
}
```


See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleClob Members](#)
- [OracleClob Constructors](#)
- [OracleClob Static Fields](#)
- [OracleClob Static Methods](#)
- [OracleClob Instance Properties](#)
- [OracleClob Instance Methods](#)

OracleClob Members

OracleClob members are listed in the following tables.

OracleClob Constructors

OracleClob constructors are listed in [Table 13-19](#).

Table 13-19 OracleClob Constructors

Constructor	Description
OracleClob Constructors	Creates an instance of the OracleClob class bound to a temporary CLOB (Overloaded)

OracleClob Static Fields

OracleClob static fields are listed in [Table 13-20](#).

Table 13-20 OracleClob Static Fields

Field	Description
MaxSize	Holds the maximum number of bytes a CLOB can hold, which is 4,294,967,295 (2 ³² - 1) bytes
Null	Represents a null value that can be assigned to the value of an OracleClob instance

OracleClob Static Methods

OracleClob static methods are listed in [Table 13-21](#).

Table 13-21 OracleClob Static Methods

Methods	Description
Equals	Inherited from System.Object (Overloaded)

OracleClob Instance Properties

OracleClob instance properties are listed in [Table 13-22](#).

Table 13-22 OracleClob Instance Properties

Properties	Description
CanRead	Indicates whether or not the LOB stream can be read
CanSeek	Indicates whether or not forward and backward seek operations can be performed
CanWrite	Indicates whether or not the LOB stream can be written
Connection	Indicates the <code>OracleConnection</code> that is used to retrieve and write CLOB data
IsEmpty	Indicates whether the CLOB is empty or not
IsInChunkWriteMode	Indicates whether or not the CLOB has been opened
IsNClob	Indicates whether or not the <code>OracleClob</code> object represents an NCLOB.
IsNull	Indicates whether or not the current instance has a null value
IsTemporary	Indicates whether or not the current instance is bound to a temporary CLOB
Length	Indicates the size of the CLOB data in bytes
OptimumChunkSize	Indicates the minimum number of bytes to retrieve or send from the database during a read or write operation
Position	Indicates the current read or write position in the LOB stream in bytes
Value	Returns the data, starting from the first character in the CLOB or NCLOB, as a string

OracleClob Instance Methods

The OracleClob instance methods are listed in [Table 13-23](#).

Table 13-23 OracleClob Instance Methods

Methods	Description
Append	Appends data to the current <code>OracleClob</code> instance (Overloaded)
BeginChunkWrite	Opens the CLOB
<code>BeginRead</code>	Inherited from <code>System.IO.Stream</code>
<code>BeginWrite</code>	Inherited from <code>System.IO.Stream</code>
Clone	Creates a copy of an <code>OracleClob</code> object
Close	Closes the current stream and releases resources associated with it
Compare	Compares data referenced by the current instance to that of the supplied object
CopyTo	Copies the data to an <code>OracleClob</code> (Overloaded)

Table 13-23 (Cont.) OracleClob Instance Methods

Methods	Description
CopyToAsync	Returns a Task-based asynchronous version of <code>OracleClob.CopyTo()</code> , which copies data from the current instance to the provided <code>OracleClob</code> object (Overloaded)
<code>CreateObjRef</code>	Inherited from <code>System.MarshalByRefObject</code>
Dispose	Releases resources allocated by this object
EndChunkWrite	Closes the CLOB referenced by the current <code>OracleClob</code> instance
<code>EndRead</code>	Inherited from <code>System.IO.Stream</code>
<code>EndWrite</code>	Inherited from <code>System.IO.Stream</code>
<code>Equals</code>	Inherited from <code>System.Object</code> (Overloaded)
Erase	Erases the specified amount of data (Overloaded)
Flush	<i>Not supported</i>
GetHashCode	Returns a hash code for the current instance
<code>GetLifetimeService</code>	Inherited from <code>System.MarshalByRefObject</code>
<code>GetType</code>	Inherited from <code>System.Object</code>
<code>InitializeLifetimeService</code>	Inherited from <code>System.MarshalByRefObject</code>
IsEqual	Compares the LOB data referenced by two <code>OracleClob</code> s
Read	Reads from the current instance (Overloaded)
ReadAsync	Returns a Task-based asynchronous version of <code>OracleClob.Read()</code> , which reads a specified number of bytes from the ODP.NET LOB instance and populates the buffer (Overloaded)
<code>ReadByte</code>	Inherited from <code>System.IO.Stream</code>
Search	Searches for a character pattern in the current instance of <code>OracleClob</code> (Overloaded)
Seek	Sets the position in the current LOB stream
SetLength	Trims or truncates the CLOB value
<code>ToString</code>	Inherited from <code>System.Object</code>
Write	Writes the provided buffer into the <code>OracleClob</code> (Overloaded)
WriteAsync	Returns a Task-based asynchronous version of <code>OracleClob.Write()</code> , which reads a specified number of bytes from the ODP.NET LOB instance and populates the buffer (Overloaded)
<code>WriteByte</code>	Inherited from <code>System.IO.Stream</code>

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)

OracleClob Constructors

OracleClob constructors create instances of the OracleClob class bound to a temporary CLOB.

Overload List:

- [OracleClob\(OracleConnection\)](#)

This constructor creates an instance of the OracleClob class bound to a temporary CLOB with an OracleConnection object.

- [OracleClob\(OracleConnection, bool, bool\)](#)

This constructor creates an instance of the OracleClob class that is bound to a temporary CLOB, with an OracleConnection object, a boolean value for caching, and a boolean value for NCLOB.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

OracleClob(OracleConnection)

This constructor creates an instance of the OracleClob class bound to a temporary CLOB with an OracleConnection object.

Declaration

```
// C#  
public OracleClob(OracleConnection con);
```

Parameters

- *con*

The OracleConnection object.

Exceptions

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

The connection must be opened explicitly by the application. `OracleClob` does not open the connection implicitly. The temporary CLOB utilizes the provided connection to store CLOB data. Caching is not enabled by default.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

OracleClob(OracleConnection, bool, bool)

This constructor creates an instance of the `OracleClob` class that is bound to a temporary CLOB, with an `OracleConnection` object, a boolean value for caching, and a boolean value for NLOB.

Declaration

```
// C#  
public OracleClob(OracleConnection con, bool bCaching, bool bNLOB);
```

Parameters

- *con*
The `OracleConnection` object connection.
- *bCaching*
A flag that indicates whether or not server-side caching is enabled.
- *bNLOB*
A flag that is set to true if the instance is a NLOB or false if it is a CLOB.

Exceptions

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The connection must be opened explicitly by the application. `OracleClob` does not open the connection implicitly. The temporary CLOB or NLOB uses the provided connection to store CLOB data.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

OracleClob Static Fields

OracleClob static fields are listed in [Table 13-24](#).

Table 13-24 OracleClob Static Fields

Field	Description
MaxSize	Holds the maximum number of bytes a CLOB can hold, which is 4,294,967,295 (2 ³² - 1) bytes
Null	Represents a null value that can be assigned to the value of an OracleClob instance

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

MaxSize

The MaxSize field holds the maximum number of bytes a CLOB can hold, which is 4,294,967,295 (2³² - 1) bytes.

Declaration

```
// C#
public static readonly Int64 MaxSize = 4294967295;
```

Remarks

This field is useful in code that checks whether or not your operation exceeds the maximum length (in bytes) allowed.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

Null

This static field represents a null value that can be assigned to the value of an `OracleClob` instance.

Declaration

```
// C#  
public static readonly OracleClob Null;
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

OracleClob Static Methods

`OracleClob` static methods are listed in [Table 13-25](#).

Table 13-25 OracleClob Static Methods

Methods	Description
Equals	Inherited from <code>System.Object</code> (Overloaded)

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

OracleClob Instance Properties

`OracleClob` instance properties are listed in [Table 13-26](#).

Table 13-26 OracleClob Instance Properties

Properties	Description
CanRead	Indicates whether or not the LOB stream can be read
CanSeek	Indicates whether or not forward and backward seek operations can be performed
CanWrite	Indicates whether or not the LOB stream can be written

Table 13-26 (Cont.) OracleClob Instance Properties

Properties	Description
Connection	Indicates the <code>OracleConnection</code> that is used to retrieve and write CLOB data
IsEmpty	Indicates whether the CLOB is empty or not
IsInChunkWriteMode	Indicates whether or not the CLOB has been opened
IsNClob	Indicates whether or not the <code>OracleClob</code> object represents an NCLOB.
IsNull	Indicates whether or not the current instance has a null value
IsTemporary	Indicates whether or not the current instance is bound to a temporary CLOB
Length	Indicates the size of the CLOB data in bytes
OptimumChunkSize	Indicates the minimum number of bytes to retrieve or send from the database during a read or write operation
Position	Indicates the current read or write position in the LOB stream in bytes
Value	Returns the data, starting from the first character in the CLOB or NCLOB, as a string

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

CanRead

Overrides `Stream`

This instance property indicates whether or not the LOB stream can be read.

Declaration

```
// C#  
public override bool CanRead{get;}
```

Property Value

If the LOB stream can be read, returns `true`; otherwise, returns `false`.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

CanSeek

Overrides Stream

This instance property indicates whether or not forward and backward seek operations can be performed.

Declaration

```
// C#  
public override bool CanSeek{get;}
```

Property Value

If forward and backward seek operations can be performed, returns true; otherwise, returns false.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

CanWrite

Overrides Stream

This instance property indicates whether or not the LOB object supports writing.

Declaration

```
// C#  
public override bool CanWrite{get;}
```

Property Value

If the LOB stream can be written, returns true; otherwise, returns false.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

Connection

This instance property indicates the OracleConnection that is used to retrieve and write CLOB data.

Declaration

```
// C#  
public OracleConnection Connection {get;}
```

Property Value

An OracleConnection.

Exceptions

ObjectDisposedException - The object is already disposed.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

IsEmpty

This instance property indicates whether the CLOB is empty or not.

Declaration

```
// C#  
public bool IsEmpty {get;}
```

Property Value

A bool.

Exceptions

ObjectDisposedException - The object is already disposed.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

IsInChunkWriteMode

This instance property indicates whether or not the CLOB has been opened to defer index updates.

Declaration

```
// C#  
public bool IsInChunkWriteMode{get;}
```

Property Value

If the CLOB has been opened, returns true; otherwise, returns false.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

IsNClob

This instance property indicates whether or not the OracleClob object represents an NClob.

Declaration

```
// C#  
public bool IsNClob {get;}
```

Property Value

A bool.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

IsNull

This property indicates whether or not the current instance has a null value.

Declaration

```
// C#  
public bool IsNull{get;}
```

Property Value

Returns true if the current instance has a null value; otherwise, returns false.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

IsTemporary

This instance property indicates whether or not the current instance is bound to a temporary CLOB.

Declaration

```
// C#  
public bool IsTemporary {get;}
```

Property Value

A bool.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

Length

Overrides Stream

This instance property indicates the size of the CLOB data in bytes.

Declaration

```
// C#  
public override Int64 Length {get;}
```

Property Value

An Int64 that indicates the size of the CLOB in bytes.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

OptimumChunkSize

This instance property indicates the minimum number of bytes to retrieve or send from the database during a read or write operation.

Declaration

```
// C#  
public int OptimumChunkSize{get;}
```

Property Value

A number representing the minimum bytes to retrieve or send.

Exceptions

ObjectDisposedException - The object is already disposed.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

Position

Overrides Stream

This instance property indicates the current read or write position in the LOB stream in bytes.

Declaration

```
// C#  
public override Int64 Position{get; set;}
```

Property Value

An Int64 that indicates the read or write position.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

ArgumentOutOfRangeException - The Position is less than 0.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

Value

This instance property returns the data, starting from the first character in the CLOB or NCLOB, as a string.

Declaration

```
// C#  
public string Value{get;}
```

Property Value

A string.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

ArgumentOutOfRangeException - The Value is less than 0.

Remarks

The value of Position is neither used nor changed by using this property.

The maximum string length that can be returned by this property is 2 GB.

For very large values, .NET may not be able to allocate enough contiguous memory to store all the data. Typically, you will see a System.OverflowException or System.OutOfMemoryException error then. To retrieve the large value, you can retrieve the data in chunks by calling GetChars() or GetBytes() method on the OracleDataReader. You can also call Read() method on the class.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

OracleClob Instance Methods

The OracleClob instance methods are listed in [Table 13-27](#).

Table 13-27 OracleClob Instance Methods

Methods	Description
Append	Appends data to the current OracleClob instance (Overloaded)
BeginChunkWrite	Opens the CLOB
BeginRead	Inherited from System.IO.Stream
BeginWrite	Inherited from System.IO.Stream
Clone	Creates a copy of an OracleClob object
Close	Closes the current stream and releases resources associated with it
Compare	Compares data referenced by the current instance to that of the supplied object
CopyTo	Copies the data to an OracleClob (Overloaded)
CopyToAsync	Returns a Task-based asynchronous version of OracleClob.CopyTo(), which copies data from the current instance to the provided OracleClob object (Overloaded)
CreateObjRef	Inherited from System.MarshalByRefObject
Dispose	Releases resources allocated by this object
EndChunkWrite	Closes the CLOB referenced by the current OracleClob instance
EndRead	Inherited from System.IO.Stream
EndWrite	Inherited from System.IO.Stream
Equals	Inherited from System.Object (Overloaded)
Erase	Erases the specified amount of data (Overloaded)
Flush	<i>Not supported</i>
GetHashCode	Returns a hash code for the current instance
GetLifetimeService	Inherited from System.MarshalByRefObject
GetType	Inherited from System.Object
InitializeLifetimeService	Inherited from System.MarshalByRefObject
IsEqual	Compares the LOB data referenced by two OracleClobs
Read	Reads from the current instance (Overloaded)
ReadAsync	Returns a Task-based asynchronous version of OracleClob.Read(), which reads a specified number of bytes from the ODP.NET LOB instance and populates the buffer (Overloaded)
ReadByte	Inherited from System.IO.Stream
Search	Searches for a character pattern in the current instance of OracleClob (Overloaded)
Seek	Sets the position in the current LOB stream

Table 13-27 (Cont.) OracleClob Instance Methods

Methods	Description
SetLength	Trims or truncates the CLOB value
ToString	Inherited from System.Object
Write	Writes the provided buffer into the OracleClob (Overloaded)
WriteAsync	Returns a Task-based asynchronous version of OracleClob.Write(), which reads a specified number of bytes from the ODP.NET LOB instance and populates the buffer (Overloaded)
WriteByte	Inherited from System.IO.Stream

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleClob Class](#)
- [OracleClob Members](#)

Append

This instance method appends data to the current OracleClob instance.

Overload List:

- [Append\(OracleClob\)](#)
This instance method appends the CLOB data referenced by the provided OracleClob object to the current OracleClob instance.
- [Append\(byte \[\], int, int\)](#)
This instance method appends data at the end of the CLOB, from the supplied byte array buffer, starting from offset (in bytes) of the supplied byte array buffer.
- [Append\(char \[\], int, int\)](#)
This instance method appends data from the supplied character array buffer to the end of the current OracleClob instance, starting at the offset (in characters) of the supplied character buffer.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleClob Class](#)
- [OracleClob Members](#)

Append(OracleClob)

This instance method appends the CLOB data referenced by the provided `OracleClob` object to the current `OracleClob` instance.

Declaration

```
// C#  
public void Append(OracleClob obj);
```

Parameters

- *obj*
An `OracleClob` object.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The parameter has a different connection than the object, `OracleConnection` is not opened, or `OracleConnection` has been reopened.

Remarks

No character set conversions are made.

The provided object and the current instance must be using the same connection; that is, the same `OracleConnection` object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

Append(byte [], int, int)

This instance method appends data at the end of the CLOB, from the supplied byte array buffer, starting from offset (in bytes) of the supplied byte array buffer.

Declaration

```
// C#  
public int Append(byte[] buffer, int offset, int count);
```

Parameters

- *buffer*
An array of bytes, representing a Unicode string.
- *offset*
The zero-based byte offset in the buffer from which data is read.

- *count*

The number of bytes to be appended.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

ArgumentOutOfRangeException - Either the *offset* or the *count* parameter is not even.

Remarks

Both *offset* and *count* must be even numbers for CLOB and NCLOB because every two bytes represent a Unicode character.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

Append(char [], int, int)

This instance method appends data from the supplied character array buffer to the end of the current OracleClob instance, starting at the offset (in characters) of the supplied character buffer.

Declaration

```
// C#  
public void Append(char[] buffer, int offset, int count);
```

Parameters

- *buffer*
An array of characters.
- *offset*
The zero-based offset (in characters) in the buffer from which data is read.
- *count*
The number of characters to be appended.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Example

```
// C#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class AppendSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleClob clob = new OracleClob(con);

        // Append 2 chars {'d', 'e'} to the OracleClob
        char[] buffer = new char[3] {'d', 'e', 'f'};
        clob.Append(buffer, 0, 2);

        // Prints "clob.Value = de"
        Console.WriteLine("clob.Value = " + clob.Value);

        clob.Close();
        clob.Dispose();

        con.Close();
        con.Dispose();
    }
}
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

BeginChunkWrite

This instance method opens the CLOB.

Declaration

```
// C#
public void BeginChunkWrite();
```

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The **OracleConnection** is not open or has been closed during the lifetime of the object.

Remarks

BeginChunkWrite does not need to be called before manipulating the CLOB data. This is provided for performance reasons.

After this method is called, write operations do not cause the domain or function-based index on the column to be updated. Index updates occur only once after EndChunkWrite is called.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

Clone

This instance method creates a copy of an OracleClob object.

Declaration

```
// C#  
public object Clone();
```

Return Value

An OracleClob object.

Implements

ICloneable

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

The cloned object has the same property values as that of the object being cloned.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Client;  
using Oracle.DataAccess.Types;  
  
class CloneSample  
{  
    static void Main()  
    {  
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
```

```
OracleConnection con = new OracleConnection(constr);
con.Open();

OracleClob clob1 = new OracleClob(con);

// Prints "clob1.Position = 0"
Console.WriteLine("clob1.Position = " + clob1.Position);

// Set the Position before calling Clone()
clob1.Position = 1;

// Clone the OracleClob
OracleClob clob2 = (OracleClob)clob1.Clone();

// Prints "clob2.Position = 1"
Console.WriteLine("clob2.Position = " + clob2.Position);

clob1.Close();
clob1.Dispose();

clob2.Close();
clob2.Dispose();

con.Close();
con.Dispose();
}
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

Close

Overrides Stream

This instance method closes the current stream and releases resources associated with it.

Declaration

```
// C#
public override void Close();
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

Compare

This instance method compares data referenced by the current instance to that of the supplied object.

Declaration

```
// C#  
public int Compare(Int64 src_offset, OracleClob obj, Int64 dst_offset,  
    Int64 amount);
```

Parameters

- *src_offset*
The comparison starting point (in characters) for the current instance.
- *obj*
The provided OracleClob object.
- *dst_offset*
The comparison starting point (in characters) for the provided OracleClob.
- *amount*
The number of characters to compare.

Return Value

The method returns a value that is:

- Less than zero: if the data referenced by the current instance is less than that of the supplied instance.
- Zero: if both objects reference the same data.
- Greater than zero: if the data referenced by the current instance is greater than that of the supplied instance.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The parameter has a different connection than the object, OracleConnection is not opened, or OracleConnection has been reopened.

ArgumentOutOfRangeException - Either the *src_offset*, *dst_offset*, or *amount* parameter is less than 0.

Remarks

The character set of the two OracleClob objects being compared should be the same for a meaningful comparison.

The provided object and the current instance must be using the same connection, that is, the same OracleConnection object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

CopyTo

CopyTo copies data from the current instance to the provided `OracleClob` object.

Overload List:

- [CopyTo\(OracleClob\)](#)
This instance method copies data from the current instance to the provided `OracleClob` object.
- [CopyTo\(OracleClob, Int64\)](#)
This instance method copies data from the current `OracleClob` instance to the provided `OracleClob` object with the specified destination offset.
- [CopyTo\(Int64, OracleClob, Int64, Int64\)](#)
This instance method copies data from the current `OracleClob` instance to the provided `OracleClob` object with the specified source offset, destination offset, and character amounts.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

CopyTo(OracleClob)

This instance method copies data from the current instance to the provided `OracleClob` object.

Declaration

```
// C#  
public Int64 CopyTo(OracleClob obj);
```

Parameters

- *obj*
The `OracleClob` object to which the data is copied.

Return Value

The return value is the amount copied.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - This exception is thrown if any of the following conditions exist:

- The `OracleConnection` is not open or has been closed during the lifetime of the object.
- The LOB object parameter has a different connection than the object.

Remarks

The provided object and the current instance must be using the same connection, that is, the same `OracleConnection` object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

CopyTo(OracleClob, Int64)

This instance method copies data from the current `OracleClob` instance to the provided `OracleClob` object with the specified destination offset.

Declaration

```
// C#  
public Int64 CopyTo(OracleClob obj, Int64 dst_offset);
```

Parameters

- *obj*
The `OracleClob` object to which the data is copied.
- *dst_offset*
The offset (in characters) at which the `OracleClob` object is copied.

Return Value

The return value is the amount copied.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentOutOfRangeException` - The *dst_offset* is less than 0.

`InvalidOperationException` - This exception is thrown if any of the following conditions exist:

- The `OracleConnection` is not open or has been closed during the lifetime of the object.
- The LOB object parameter has a different connection than the object.

Remarks

If the *dst_offset* is beyond the end of the `OracleClob` data, spaces are written into the `OracleClob` until the *dst_offset* is met.

The offsets are 0-based. No character conversion is performed by this operation.

The provided object and the current instance must be using the same connection; that is, the same `OracleConnection` object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

CopyTo(Int64, OracleClob, Int64, Int64)

This instance method copies data from the current `OracleClob` instance to the provided `OracleClob` object with the specified source offset, destination offset, and character amounts.

Declaration

```
// C#
public Int64 CopyTo(Int64 src_offset, OracleClob obj, Int64 dst_offset,
    Int64 amount);
```

Parameters

- *src_offset*
The offset (in characters) in the current instance, from which the data is read.
- *obj*
The `OracleClob` object to which the data is copied.
- *dst_offset*
The offset (in characters) at which the `OracleClob` object is copied.
- *amount*
The amount of data to be copied.

Return Value

The return value is the amount copied.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The parameter has a different connection than the object, `OracleConnection` is not opened, or `OracleConnection` has been reopened.

`ArgumentOutOfRangeException` - The *src_offset*, the *dst_offset*, or the *amount* parameter is less than 0.

Remarks

If the *dst_offset* is beyond the end of the `OracleClob` data, spaces are written into the `OracleClob` until the *dst_offset* is met.

The offsets are 0-based. No character conversion is performed by this operation.

The provided object and the current instance must be using the same connection, that is, the same `OracleConnection` object.

Example

```
// C#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class CopyToSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleClob clob1 = new OracleClob(con);
        OracleClob clob2 = new OracleClob(con);

        // Write 4 chars, starting at buffer offset 0
        char[] buffer = new char[4] {'a', 'b', 'c', 'd'};
        clob1.Write(buffer, 0, 4);

        // Copy 2 chars from char 0 of clob1 to char 1 of clob2
        clob1.CopyTo(0, clob2, 1, 2);

        //Prints "clob2.Value = ab"
        Console.WriteLine("clob2.Value = " + clob2.Value);

        clob1.Close();
        clob1.Dispose();

        clob2.Close();
        clob2.Dispose();

        con.Close();
        con.Dispose();
    }
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

CopyToAsync

CopyToAsync returns a Task-based asynchronous version of `OracleClob.CopyTo()`, which copies data from the current instance to the provided `OracleClob` object.

Overload List:

- [CopyToAsync\(Int64, OracleClob, Int64, Int64\)](#)
This method returns a Task-based asynchronous version of `OracleClob.CopyTo()`, which copies data from the current instance to the provided `OracleClob` object.
- [CopyToAsync\(Int64, OracleClob, Int64, Int64, CancellationToken\)](#)
This method returns a Task-based asynchronous version of `OracleClob.CopyTo()`, which copies data from the current instance to the provided `OracleClob` object.
- [CopyToAsync\(OracleClob, Int64\)](#)
This method returns a Task-based asynchronous version of `OracleClob.CopyTo()`, which copies data from the current instance to the provided `OracleClob` object.
- [CopyToAsync\(OracleClob, Int64, CancellationToken\)](#)
This method returns a Task-based asynchronous version of `OracleClob.CopyTo()`, which copies data from the current instance to the provided `OracleClob` object.
- [CopyToAsync\(OracleClob\)](#)
This method returns a Task-based asynchronous version of `OracleClob.CopyTo()`, which copies data from the current instance to the provided `OracleClob` object.
- [CopyToAsync\(OracleClob, CancellationToken\)](#)
This method returns a Task-based asynchronous version of `OracleClob.CopyTo()`, which copies data from the current instance to the provided `OracleClob` object.

Example

```
using Oracle.ManagedDataAccess.Client;
using Oracle.ManagedDataAccess.Types;
using System;
using System.Threading;

namespace AsyncApp
{
    class AsyncDemo
    {
        static async Task Main(string[] args)
        {
            string connectionString = "User Id=HR; Password=<PASSWORD>; Data Source=oracle;";
```

```

OracleConnection oc = new OracleConnection(connectionString);
await oc.OpenAsync(CancellationToken.None);

OracleCommand cmd = oc.CreateCommand();
cmd.CommandText = " select clob_column from tab1";

OracleClob clob1 = new OracleClob(oc);
OracleClob clob2 = new OracleClob(oc);
OracleClob clob3 = new OracleClob(oc);
OracleClob clob4 = new OracleClob(oc);
OracleClob clob5 = new OracleClob(oc);
OracleClob clob6 = new OracleClob(oc);

OracleDataReader reader = await cmd.ExecuteReaderAsync();

await reader.ReadAsync(CancellationToken.None);
using (OracleClob clob = reader.GetOracleClob(0))
{
    //asynchronously copy clob data
    Int64 charsCopied1 = await clob.CopyToAsync(0, clob1, 0, clob.Length);
    Console.WriteLine("chars copied to clob1 = " + charsCopied1);

    //asynchronously copy clob data
    Int64 charsCopied2 = await clob.CopyToAsync(0, clob2, 0, clob.Length,
CancellationToken.None);
    Console.WriteLine("chars copied to clob2 = " + charsCopied2);

    //asynchronously copy clob data
    Int64 charsCopied3 = await clob.CopyToAsync(clob3, 0);
    Console.WriteLine("chars copied to clob3 = " + charsCopied3);

    //asynchronously copy clob data
    Int64 charsCopied4 = await clob.CopyToAsync(clob4, 0, CancellationToken.None);
    Console.WriteLine("chars copied to clob4 = " + charsCopied4);

    //asynchronously copy clob data
    Int64 charsCopied5 = await clob.CopyToAsync(clob5);
    Console.WriteLine("chars copied to clob5 = " + charsCopied5);

    //asynchronously copy clob data
    Int64 charsCopied6 = await clob.CopyToAsync(clob6, CancellationToken.None);
    Console.WriteLine("chars copied to clob6 = " + charsCopied6);
}
}
}
}

```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

CopyToAsync(Int64, OracleClob, Int64, Int64)

This method returns a Task-based asynchronous version of `OracleClob.CopyTo()`, which copies data from the current instance to the provided `OracleClob` object.

Declaration

```
// C#  
public Task<Int64> CopyToAsync(Int64 src_offset, OracleClob obj, Int64 dst_offset, Int64  
amount)
```

Parameters

- *src_offset*
The offset (in characters) in the current instance, from which the data is read.
- *obj*
The OracleClob object to which the data is copied.
- *dst_offset*
The offset (in characters) at which the OracleClob object is copied.
- *amount*
The amount of data to be copied.

Return Value

Task<Int64> object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `ArgumentOutOfRangeException` - The *src_offset*, the *dst_offset*, or the *amount* parameter is less than 0.
- `InvalidOperationException` - This exception is thrown if any of the following conditions exist:
 - The `OracleConnection` is not open or has been closed during the lifetime of the object.
 - The LOB object parameter has a different connection than the object.

📘 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

CopyToAsync(Int64, OracleClob, Int64, Int64, CancellationToken)

This method returns a Task-based asynchronous version of `OracleClob.CopyTo()`, which copies data from the current instance to the provided `OracleClob` object.

Declaration

```
// C#  
public Task<Int64> CopyToAsync(Int64 src_offset, OracleClob obj, Int64 dst_offset, Int64  
amount, CancellationToken cancellationToken)
```

Parameters

- *src_offset*
The offset (in characters) in the current instance, from which the data is read.
- *obj*
The OracleClob object to which the data is copied.
- *dst_offset*
The offset (in characters) at which the OracleClob object is copied.
- *amount*
The amount of data to be copied.
- *cancellationToken*
The input cancellation token which can be used by the application to cancel the task.

Return Value

Task<Int64> object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `ArgumentOutOfRangeException` - The *src_offset*, the *dst_offset*, or the *amount* parameter is less than 0.
- `InvalidOperationException` - This exception is thrown if any of the following conditions exist:
 - The `OracleConnection` is not open or has been closed during the lifetime of the object.
 - The LOB object parameter has a different connection than the object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

CopyToAsync(OracleClob, Int64)

This method returns a Task-based asynchronous version of `OracleClob.CopyTo()`, which copies data from the current instance to the provided `OracleClob` object.

Declaration

```
// C#  
public async Task<Int64> CopyToAsync(OracleClob obj, Int64 dst_offset)
```

Parameters

- *obj*

The `OracleClob` object to which the data is copied.

- `dst_offset`

The offset (in characters) at which the `OracleClob` object is copied.

Return Value

`Task<Int64>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `ArgumentOutOfRangeException` - The `dst_offset` is less than 0.
- `InvalidOperationException` - This exception is thrown if any of the following conditions exist:
 - The `OracleConnection` is not open or has been closed during the lifetime of the object.
 - The LOB object parameter has a different connection than the object.

❗ See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

CopyToAsync(OracleClob, Int64, CancellationTokens)

This method returns a Task-based asynchronous version of `OracleClob.CopyTo()`, which copies data from the current instance to the provided `OracleClob` object.

Declaration

```
// C#
public async Task<Int64> CopyToAsync(OracleClob obj, Int64 dst_offset, CancellationTokens cancellationTokens)
```

Parameters

- `obj`
The `OracleClob` object to which the data is copied.
- `dst_offset`
The offset (in characters) at which the `OracleClob` object is copied.
- `cancellationTokens`
The input cancellation token which can be used by the application to cancel the task.

Return Value

`Task<Int64>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `ArgumentOutOfRangeException` - The `dst_offset` parameter is less than 0.
- `InvalidOperationException` - This exception is thrown if any of the following conditions exist:
 - The `OracleConnection` is not open or has been closed during the lifetime of the object.
 - The LOB object parameter has a different connection than the object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

CopyToAsync(OracleClob)

This method returns a Task-based asynchronous version of `OracleClob.CopyTo()`, which copies data from the current instance to the provided `OracleClob` object.

Declaration

```
// C#  
public async Task<Int64> CopyToAsync(OracleClob obj)
```

Parameters

- *obj*
The `OracleClob` object to which the data is copied.

Return Value

`Task<Int64>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `InvalidOperationException` - This exception is thrown if any of the following conditions exist:
 - The `OracleConnection` is not open or has been closed during the lifetime of the object.
 - The LOB object parameter has a different connection than the object.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

CopyToAsync(OracleClob, CancellationToken)

This method returns a Task-based asynchronous version of `OracleClob.CopyTo()`, which copies data from the current instance to the provided `OracleClob` object.

Declaration

```
// C#  
public async Task<Int64> CopyToAsync(OracleClob obj, CancellationToken cancellationToken)
```

Parameters

- *obj*
The `OracleClob` object to which the data is copied.
- *cancellationToken*
The input cancellation token which can be used by the application to cancel the task.

Return Value

`Task<Int64>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `InvalidOperationException` - This exception is thrown if any of the following conditions exist:
 - The `OracleConnection` is not open or has been closed during the lifetime of the object.
 - The LOB object parameter has a different connection than the object.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

Dispose

This instance method releases resources allocated by this object.

Declaration

```
public void Dispose();
```

Implements

IDisposable

Remarks

The object cannot be reused after being disposed. Although some properties can still be accessed, their values cannot be accountable. Since resources are freed, method calls can lead to exceptions.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

EndChunkWrite

This instance method closes the CLOB referenced by the current OracleClob instance.

Declaration

```
// C#  
public void EndChunkWrite();
```

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

Index updates occur immediately if write operation(s) are deferred by the BeginChunkWrite method.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

Erase

Erase erases part or all data.

Overload List:

- [Erase\(\)](#)

This instance method erases all data.

- [Erase\(Int64, Int64\)](#)

This instance method replaces the specified amount of data (in characters) starting from the specified offset with zero-byte fillers (in characters).

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

Erase()

This instance method erases all data.

Declaration

```
// C#  
public Int64 Erase();
```

Return Value

The number of characters erased.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

Erase(Int64, Int64)

This instance method replaces the specified amount of data (in characters) starting from the specified offset with zero-byte fillers (in characters).

Declaration

```
// C#  
public Int64 Erase(Int64 offset, Int64 amount);
```

Parameters

- *offset*
The offset.
- *amount*
The amount of data.

Return Value

The actual number of characters erased.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

ArgumentOutOfRangeException - The *offset* or *amount* parameter is less than 0.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

Flush

This method is not supported.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

GetHashCode

Overrides Object

This method returns a hash code for the current instance.

Declaration

```
// C#  
public override int GetHashCode();
```

Return Value

An int representing a hash code.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

IsEqual

This instance method compares the LOB data referenced by two OracleClob.

Declaration

```
// C#  
public bool IsEqual(OracleClob obj);
```

Parameters

- *obj*
An OracleClob object.

Return Value

Returns true if the current OracleClob and the provided OracleClob refer to the same LOB. Otherwise, returns false.

Remarks

Note that this method can return true even if the two OracleClob objects returns false for `==` or `Equals()` because two different OracleClob instances can refer to the same LOB.

The provided object and the current instance must be using the same connection, that is, the same OracleConnection object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

Read

Read reads a specified amount from the current instance and populates the array buffer.

Overload List:

- [Read\(byte \[\], int, int\)](#)
This instance method reads a specified amount of bytes from the current instance and populates the byte array buffer.
- [Read\(char \[\], int, int\)](#)
This instance method reads a specified amount of characters from the current instance and populates the character array buffer.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

Read(byte [], int, int)

Overrides Stream

This instance method reads a specified amount of bytes from the current instance and populates the byte array buffer.

Declaration

```
// C#  
public override int Read(byte [] buffer, int offset, int count);
```

Parameters

- *buffer*
The byte array buffer that is populated.
- *offset*
The offset (in bytes) at which the buffer is populated.
- *count*
The amount of bytes to be read.

Return Value

The number of bytes read from the CLOB.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

Both *offset* and *count* must be even numbers for CLOB and NCLOB because every two bytes represent a Unicode character.

The LOB data is read starting from the position specified by the *Position* property, which must also be an even number.

OracleClob is free to return fewer bytes than requested, even if the end of the stream has not been reached.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

Read(char [], int, int)

This instance method reads a specified amount of characters from the current instance and populates the character array buffer.

Declaration

```
// C#  
public int Read(char[ ] buffer, int offset, int count);
```

Parameters

- *buffer*
The character array buffer that is populated.
- *offset*
The offset (in characters) at which the buffer is populated.
- *count*
The amount of characters to be read.

Return Value

The return value indicates the number of characters read from the CLOB.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The *OracleConnection* is not open or has been closed during the lifetime of the object.

ArgumentOutOfRangeException - This exception is thrown if any of the following conditions exist:

- The *offset* or the *count* is less than 0.

- The *offset* is greater than or equal to the *buffer.Length*.
- The *offset* and the *count* together are greater than *buffer.Length*.

Remarks

Handles all CLOB and NCLOB data as Unicode.

The LOB data is read starting from the position specified by the *Position* property.

Example

```
// C#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class ReadSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleClob clob = new OracleClob(con);

        // Write 3 chars, starting at buffer offset 1
        char[] writeBuffer = new char[4] {'a', 'b', 'c', 'd'};
        clob.Write(writeBuffer, 1, 3);

        // Reset the Position (in bytes) for Read
        clob.Position = 2;

        // Read 2 chars into buffer starting at buffer offset 1
        char[] readBuffer = new char[4];
        int charsRead = clob.Read(readBuffer, 1, 2);

        // Prints "charsRead = 2"
        Console.WriteLine("charsRead = " + charsRead);

        // Prints "readBuffer = cd "
        Console.Write("readBuffer = ");
        for(int index = 0; index < readBuffer.Length; index++)
        {
            Console.Write(readBuffer[index]);
        }
        Console.WriteLine();

        clob.Close();
        clob.Dispose();

        con.Close();
        con.Dispose();
    }
}
```


① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

ReadAsync

ReadAsync returns a Task-based asynchronous version of `OracleClob.Read()`, which reads a specified number of bytes from the ODP.NET LOB instance and populates the buffer.

Overload List:

- [ReadAsync\(byte\[\], int, int, CancellationToken\)](#)
ReadAsync returns a Task-based asynchronous version of `OracleClob.Read()`, which reads a specified number of bytes from the ODP.NET LOB instance and populates the buffer.
- [ReadAsync\(byte\[\], int, int\)](#)
ReadAsync returns a Task-based asynchronous version of `OracleClob.Read()`, which reads a specified number of bytes from the ODP.NET LOB instance and populates the buffer.
- [ReadAsync\(char\[\], int, int\)](#)
ReadAsync returns a Task-based asynchronous version of `OracleClob.Read()`, which reads a specified number of bytes from the ODP.NET LOB instance and populates the buffer.
- [ReadAsync\(char\[\], int, int, CancellationToken\)](#)
ReadAsync returns a Task-based asynchronous version of `OracleClob.Read()`, which reads a specified number of bytes from the ODP.NET LOB instance and populates the buffer.

Example (includes all overloads)

```
using Oracle.ManagedDataAccess.Client;
using Oracle.ManagedDataAccess.Types;
using System;
using System.Threading;
using System.Threading.Tasks;

namespace AsyncApp
{
    class AsyncDemo
    {
        static async Task Main(string[] args)
        {
            string connectionString = "User Id=HR; Password=<PASSWORD>; Data Source=oracle;";

            OracleConnection oc = new OracleConnection(connectionString);
            await oc.OpenAsync(CancellationToken.None);

            OracleCommand cmd = oc.CreateCommand();
```

```

cmd.CommandText = " select clob_column from tab1";

OracleDataReader reader = await cmd.ExecuteReaderAsync();

await reader.ReadAsync(CancellationToken.None);

using (var clob = reader.GetOracleClob(0))
{
    char[] charBuffer = new char[128];
    byte[] byteBuffer = new byte[128];

    //asynchronously read clob data
    int bytesRead = await clob.ReadAsync(byteBuffer, 0, 64);
    Console.WriteLine("Bytes Read: " + bytesRead);

    //asynchronously read clob data
    bytesRead = await clob.ReadAsync(byteBuffer, 64, 64, CancellationToken.None);
    Console.WriteLine("Bytes Read: " + bytesRead);

    //asynchronously read clob data
    int charsRead = await clob.ReadAsync(charBuffer, 0, 64);
    Console.WriteLine("Chars Read: " + charsRead);

    //asynchronously read clob data
    charsRead = await clob.ReadAsync(charBuffer, 64, 64, CancellationToken.None);
    Console.WriteLine("Chars Read: " + charsRead);
}
}
}
}

```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

ReadAsync(byte[], int, int, CancellationToken)

This method returns a Task-based asynchronous version of `OracleClob.Read()`, which reads a specified number of bytes from the ODP.NET LOB instance and populates the buffer.

Declaration

```

// C#
public override Task<int> ReadAsync(byte[] buffer, int offset, int count,
CancellationToken cancellationToken)

```

Parameters

- *buffer*
The byte array buffer that is populated.
- *offset*
The offset (in bytes) at which the buffer is populated.

- *count*
The amount of bytes to be read.
- *cancellationToken*
The input cancellation token which can be used by the application to cancel the task.

Return Value

Task<int> object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Implements

Stream

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.
- `ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:
 - The offset or the count parameter is less than 0.
 - The offset is greater than or equal to the `buffer.Length`.
 - The offset and the count together are greater than the `buffer.Length`.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

ReadAsync(byte[], int, int)

This method calls into the `ReadAsync` implementation with argument 'cancellationToken' passed as `Cancellation.Token.None`.

Declaration

```
// C#  
public Task<int> ReadAsync(byte[] buffer, int offset, int count)
```

Parameters

- *buffer*
The byte array buffer that is populated.
- *offset*
The offset (in bytes) at which the buffer is populated.
- *count*

The amount of bytes to be read.

Return Value

`Task<int>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Implements

Stream

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.
- `ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:
 - The offset or the count parameter is less than 0.
 - The offset is greater than or equal to the `buffer.Length`.
 - The offset and the count together are greater than the `buffer.Length`.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

ReadAsync(char[], int, int)

This method calls into the `ReadAsync` implementation with argument 'cancellationToken' passed as `CancellationToken.None`.

Declaration

```
// C#  
public Task<int> ReadAsync(char[] buffer, int offset, int count)
```

Parameters

- *buffer*
The character array buffer that is populated.
- *offset*
The offset (in characters) at which the buffer is populated.
- *count*
The amount of characters to be read.

Return Value

`Task<int>` object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.
- `ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:
 - The offset or the count parameter is less than 0.
 - The offset is greater than or equal to the `buffer.Length`.
 - The offset and the count together are greater than the `buffer.Length`.

📘 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

ReadAsync(char[], int, int, CancellationToken)

This method returns a Task-based asynchronous version of `OracleClob.Read()`, which reads a specified number of bytes from the ODP.NET LOB instance and populates the buffer.

Declaration

```
// C#  
public Task<int> ReadAsync(char[] buffer, int offset, int count, CancellationToken  
cancellationToken)
```

Parameters

- *buffer*
The character array buffer that is populated.
- *offset*
The offset (in characters) at which the buffer is populated.
- *count*
The amount of characters to be read.
- *cancellationToken*
The input cancellation token which can be used by the application to cancel the task.

Return Value

Task<int> object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.
- `ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:
 - The offset or the count parameter is less than 0.
 - The offset is greater than or equal to the `buffer.Length`.
 - The offset and the count together are greater than the `buffer.Length`.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

Search

Search searches for a character pattern in the current instance of `OracleClob`.

Overload List:

- [Search\(byte\[\], Int64, Int64\)](#)
This instance method searches for a character pattern, represented by the byte array, in the current instance of `OracleClob`.
- [Search\(char\[\], Int64, Int64\)](#)
This instance method searches for a character pattern in the current instance of `OracleClob`.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

Search(byte[], Int64, Int64)

This instance method searches for a character pattern, represented by the byte array, in the current instance of `OracleClob`.

Declaration

```
// C#  
public int Search(byte[] val, Int64 offset, Int64 nth);
```

Parameters

- *val*
A Unicode byte array.
- *offset*
The 0-based offset (in characters) starting from which the `OracleClob` is searched.
- *nth*
The specific occurrence (1-based) of the match for which the absolute offset (in characters) is returned.

Return Value

Returns the absolute *offset* of the start of the matched pattern (in bytes) for the *nth* occurrence of the match. Otherwise, 0 is returned.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:

- The *offset* is less than 0.
- The *nth* is less than or equal to 0.
- The *nth* is greater than or equal to `OracleClob.MaxValue`.
- The *offset* is greater than or equal to `OracleClob.MaxValue`.

Remarks

The `byte[ ]` is converted to Unicode before the search is made.

The limit of the search pattern is 16383 bytes.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

Search(char[], Int64, Int64)

This instance method searches for a character pattern in the current instance of `OracleClob`.

Declaration

```
// C#  
public Int64 Search(char [ ] val, Int64 offset, Int64 nth);
```

Parameters

- *val*
The Unicode string being searched for.
- *offset*
The 0-based offset (in characters) starting from which the `OracleClob` is searched.
- *nth*
The specific occurrence (1-based) of the match for which the absolute offset (in characters) is returned.

Return Value

Returns the absolute *offset* of the start of the matched pattern (in characters) for the *nth* occurrence of the match. Otherwise, 0 is returned.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:

- The *offset* is less than 0.
- The *nth* is less than or equal to 0.
- The *val*.Length doubled is greater than 16383.
- The *nth* is greater than or equal to `OracleClob.MaxValue`.
- The *offset* is greater than or equal to `OracleClob.MaxValue`.

Remarks

The limit of the search pattern is 16383 bytes.

Example

```
// C#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class SearchSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleClob clob = new OracleClob(con);

        // Write 7 chars, starting at buffer offset 0
        char[] buffer = new char[7] {'a', 'b', 'c', 'd', 'a', 'b', 'c'};
        clob.Write(buffer, 0, 7);

        // Search for the 2nd occurrence of a char pattern 'bc'
        // starting at offset 1 in the OracleBlob
        char[] pattern = new char[2] {'b', 'c'};
        long posFound = clob.Search(pattern, 1, 2);

        // Prints "posFound = 6"
        Console.WriteLine("posFound = " + posFound);

        clob.Close();
        clob.Dispose();

        con.Close();
        con.Dispose();
    }
}
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

Seek

Overrides Stream

This instance method sets the position on the current LOB stream.

Declaration

```
// C#
public override Int64 Seek(Int64 offset, SeekOrigin origin);
```

Parameters

- *offset*
A byte offset relative to origin.
- *origin*
A value of type `System.IO.SeekOrigin` indicating the reference point used to obtain the new position.

Return Value

Returns an `Int64` that indicates the position.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

If *offset* is negative, the new position precedes the position specified by *origin* by the number of characters specified by *offset*.

If *offset* is zero, the new position is the position specified by *origin*.

If *offset* is positive, the new position follows the position specified by *origin* by the number of characters specified by *offset*.

`SeekOrigin.Begin` specifies the beginning of a stream.

`SeekOrigin.Current` specifies the current position within a stream.

`SeekOrigin.End` specifies the end of a stream.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

SetLength

Overrides `Stream`

This instance method trims or truncates the CLOB value to the specified length (in characters).

Declaration

```
// C#  
public override void SetLength(Int64 newlen);
```

Parameters

- *newlen*
The desired length of the current stream in characters.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The `OracleConnection` is not open or has been closed during the lifetime of the object.

ArgumentOutOfRangeException - The *newlen* parameter is greater than 0.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

Write

This instance method writes data from the provided array buffer into the `OracleClob`.

Overload List:

- [Write\(byte\[\], int, int\)](#)
This instance method writes data from the provided byte array buffer into the `OracleClob`.
- [Write\(char\[\], int, int\)](#)
This instance method writes data from the provided character array buffer into the `OracleClob`.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

WriteAsync

`WriteAsync` returns a Task-based asynchronous version of `OracleClob.Write()`, which reads a specified number of bytes from the ODP.NET LOB instance and populates the buffer.

Overload List:

- [WriteAsync\(byte\[\], int, int\)](#)

WriteAsync returns a Task-based asynchronous version of OracleClob.Write(), which reads a specified number of bytes from the ODP.NET LOB instance and populates the buffer.

- [WriteAsync\(byte\[\], int, int, CancellationToken\)](#)

WriteAsync returns a Task-based asynchronous version of OracleClob.Write(), which reads a specified number of bytes from the ODP.NET LOB instance and populates the buffer.

- [WriteAsync\(char\[\], int, int\)](#)

WriteAsync returns a Task-based asynchronous version of OracleClob.Write(), which reads a specified number of bytes from the ODP.NET LOB instance and populates the buffer.

- [WriteAsync\(char\[\], int, int, CancellationToken\)](#)

WriteAsync returns a Task-based asynchronous version of OracleClob.Write(), which reads a specified number of bytes from the ODP.NET LOB instance and populates the buffer.

Example (includes all overloads)

```
using Oracle.ManagedDataAccess.Client;
using Oracle.ManagedDataAccess.Types;
using System;
using System.Threading;
using System.Threading.Tasks;

namespace AsyncApp
{
    class AsyncDemo
    {
        static async Task Main(string[] args)
        {
            string connectionString = "User Id=HR; Password=<PASSWORD>; Data Source=oracle;";

            OracleConnection oc = new OracleConnection(connectionString);
            await oc.OpenAsync(CancellationToken.None);

            OracleClob clob = new OracleClob(oc);

            char[] writeBufferChar = new char[4] { 'a', 'b', 'c', 'd' };
            byte[] writeBufferByte = new byte[4] { 1, 2, 3, 4 };

            //Write data to Character Large Object, asynchronously
            await clob.WriteAsync(writeBufferByte, 0, 4);
            Console.WriteLine("clob.Length = " + clob.Length);

            //Write data to Character Large Object, asynchronously
            await clob.WriteAsync(writeBufferByte, 0, 4, CancellationToken.None);
            Console.WriteLine("clob.Length = " + clob.Length);

            //Write data to Character Large Object, asynchronously
            await clob.WriteAsync(writeBufferChar, 0, 4);
            Console.WriteLine("clob.Length = " + clob.Length);

            //Write data to Character Large Object, asynchronously
            await clob.WriteAsync(writeBufferChar, 0, 4, CancellationToken.None);
            Console.WriteLine("clob.Length = " + clob.Length);
        }
    }
}
```

```
}  
}
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

WriteAsync(byte[], int, int)

This method calls into the WriteAsync implementation with argument 'cancellationToken' passed as CancellationToken.None.

Declaration

```
// C#  
public Task WriteAsync(byte[] buffer, int offset, int count)
```

Parameters

- *buffer*
The byte array buffer that provides the data.
- *offset*
The 0-based offset (in bytes) from which the buffer is read.
- *count*
The amount of bytes to be written into the OracleClob.

Return Value

Task object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Implements

Stream

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.
- `ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:
 - The offset or the count parameter is less than 0.
 - The offset is greater than or equal to the `buffer.Length`.
 - The offset and the count together are greater than the `buffer.Length`.
 - The offset, the count, or the `Position` is not even.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

WriteAsync(byte[], int, int, CancellationToken)

This method returns a Task-based asynchronous version of `OracleClob.Write()`, which writes a specified number of bytes from the ODP.NET LOB instance and populates the buffer.

Declaration

```
// C#  
public override Task WriteAsync(byte[] buffer, int offset, int count, CancellationToken  
cancellationTokentoken)
```

Parameters

- *buffer*
The byte array buffer that provides the data.
- *offset*
The 0-based offset (in bytes) from which the buffer is read.
- *count*
The amount of bytes to be written into the `OracleClob`.
- *cancellationTokentoken*
The input cancellation token which can be used by the application to cancel the task.

Return Value

Task object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Implements

Stream

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.
- `ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:
 - The offset or the count parameter is less than 0.
 - The offset is greater than or equal to the `buffer.Length`.
 - The offset and the count together are greater than the `buffer.Length`.
 - The offset, the count, or the `Position` is not even.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

WriteAsync(char[], int, int)

This method calls into the WriteAsync implementation with argument 'cancellationToken' passed as CancellationToken.None.

Declaration

```
// C#  
public Task WriteAsync(char[] buffer, int offset, int count)
```

Parameters

- *buffer*
The character array buffer that provides the data.
- *offset*
The 0-based offset (in characters) from which the buffer is read.
- *count*
The amount of characters to be written into the OracleClob.

Return Value

Task object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.
- `ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:
 - The offset or the count parameter is less than 0.
 - The offset is greater than or equal to the `buffer.Length`.
 - The offset and the count together are greater than the `buffer.Length`.
 - The offset, the count, or the `Position` is not even.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

WriteAsync(char[], int, int, CancellationToken)

This method returns a Task-based asynchronous version of `OracleClob.Write()`, which writes a specified number of bytes from the ODP.NET LOB instance and populates the buffer.

Declaration

```
// C#  
public Task WriteAsync(char[] buffer, int offset, int count, CancellationToken  
cancellationTokentoken)
```

Parameters

- *buffer*
The character array buffer that provides the data.
- *offset*
The 0-based offset (in characters) from which the buffer is read.
- *count*
The amount of characters to be written into the `OracleClob`.
- *cancellationTokentoken*
The input cancellation token which can be used by the application to cancel the task.

Return Value

Task object representing the asynchronous operation immediately without blocking the calling thread for the whole duration of the query execution.

Exceptions

- `ObjectDisposedException` - The object is already disposed.
- `InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.
- `ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:
 - The offset or the count parameter is less than 0.
 - The offset is greater than or equal to the `buffer.Length`.
 - The offset and the count together are greater than the `buffer.Length`.
 - The offset, the count, or the `Position` is not even.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

Write(byte[], int, int)

Overrides Stream

This instance method writes data from the provided byte array buffer into the OracleClob.

Declaration

```
// C#  
public override void Write(byte[ ] buffer, int offset, int count);
```

Parameters

- *buffer*
The byte array buffer that represents a Unicode string.
- *offset*
The offset (in bytes) from which the buffer is read.
- *count*
The amount of data (in bytes) from the buffer to be written into the OracleClob.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

ArgumentOutOfRangeException - This exception is thrown if any of the following conditions exist:

- The *offset* or the *count* is less than 0.
- The *offset* is greater than or equal to the *buffer*.Length.
- The *offset* and the *count* together are greater than the *buffer*.Length.
- The *offset*, the *count*, or the *Position* is not even.

Remarks

Both *offset* and *count* must be even numbers for CLOB and NCLOB because every two bytes represent a Unicode character.

The LOB data is read starting from the position specified by the *Position* property. The *Position* property must be an even number.

If necessary, proper data conversion is carried out from the client character set to the database character set.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

Write(char[], int, int)

This instance method writes data from the provided character array buffer into the `OracleClob`.

Declaration

```
// C#  
public void Write(char[] buffer, int offset, int count);
```

Parameters

- *buffer*
The character array buffer that is written to the `OracleClob`.
- *offset*
The offset (in characters) from which the *buffer* is read.
- *count*
The amount (in characters) from the buffer that is to be written into the `OracleClob`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:

- The *offset* or the *count* is less than 0.
- The *offset* is greater than or equal to the *buffer.Length*.
- The *offset* and the *count* together are greater than *buffer.Length*.
- The Position is not even.

Remarks

Handles all CLOB and NCLOB data as Unicode.

The LOB data is read starting from the position specified by the `Position` property.

If necessary, proper data conversion is carried out from the client character set to the database character set.

Example

```
// C#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class WriteSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleClob clob = new OracleClob(con);

        // Set the Position for the Write;
        clob.Position = 0;

        // Begin ChunkWrite to improve performance
        // Index updates occur only once after EndChunkWrite
        clob.BeginChunkWrite();

        // Write to the OracleClob in 5 chunks of 2 chars each
        char[] c = new char[2] {'a', 'b'};
        for (int index = 0; index < 5; index++)
        {
            clob.Write(c, 0, c.Length);
        }
        clob.EndChunkWrite();

        // Prints "clob.Value = ababababab"
        Console.WriteLine("clob.Value = " + clob.Value);

        clob.Close();
        clob.Dispose();

        con.Close();
        con.Dispose();
    }
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleClob Class](#)
- [OracleClob Members](#)

OracleRefCursor Class

An OracleRefCursor object represents an Oracle REF CURSOR..

Class Inheritance

System.Object
 System.MarshalRefByObject
 Oracle.DataAccess.Types.OracleRefCursor

Declaration

```
// C#  
public sealed class OracleRefCursor : MarshalByRefObject, IDisposable, INullable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

To minimize the number of open server cursors, OracleRefReader objects should be explicitly disposed.

Example

```
// Database Setup  
/*  
connect scott/tiger@oracle  
CREATE OR REPLACE FUNCTION MyFunc(refcur_out OUT SYS_REFCURSOR)  
    RETURN SYS_REFCURSOR IS refcur_ret SYS_REFCURSOR;  
BEGIN  
    OPEN refcur_ret FOR SELECT * FROM EMP;  
    OPEN refcur_out FOR SELECT * FROM DEPT;  
    RETURN refcur_ret;  
END MyFunc;  
/  
*/  
  
// C#  
  
using System;  
using System.Data;  
using Oracle.DataAccess.Client;  
using Oracle.DataAccess.Types;
```

```
class OracleRefCursorSample
{
    static void Main()
    {
        // Example demonstrates how to use REF CURSORS returned from
        // PL/SQL Stored Procedures or Functions
        // Create the PL/SQL Function MyFunc as defined previously

        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        // Create an OracleCommand
        OracleCommand cmd = new OracleCommand("MyFunc", con);
        cmd.CommandType = CommandType.StoredProcedure;

        // Bind the parameters
        // p1 is the RETURN REF CURSOR bound to SELECT * FROM EMP;
        OracleParameter p1 =
            cmd.Parameters.Add("refcur_ret", OracleDbType.RefCursor);
        p1.Direction = ParameterDirection.ReturnValue;

        // p2 is the OUT REF CURSOR bound to SELECT * FROM DEPT
        OracleParameter p2 =
            cmd.Parameters.Add("refcur_out", OracleDbType.RefCursor);
        p2.Direction = ParameterDirection.Output;

        // Execute the command
        cmd.ExecuteNonQuery();

        // Construct an OracleDataReader from the REF CURSOR
        OracleDataReader reader1 = ((OracleRefCursor)p1.Value).GetDataReader();

        // Prints "reader1.GetName(0) = EMPNO"
        Console.WriteLine("reader1.GetName(0) = " + reader1.GetName(0));

        // Construct an OracleDataReader from the REF CURSOR
        OracleDataReader reader2 = ((OracleRefCursor)p2.Value).GetDataReader();

        // Prints "reader2.GetName(0) = DEPTNO"
        Console.WriteLine("reader2.GetName(0) = " + reader2.GetName(0));

        reader1.Close();
        reader1.Dispose();

        reader2.Close();
        reader2.Dispose();

        p1.Dispose();
        p2.Dispose();

        cmd.Dispose();

        con.Close();
        con.Dispose();
    }
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRefCursor Members](#)
- [OracleRefCursor Static Methods](#)
- [OracleRefCursor Static Fields](#)
- [OracleRefCursor Properties](#)
- [OracleRefCursor Instance Methods](#)

OracleRefCursor Members

OracleRefCursor members are listed in the following tables.

OracleRefCursor Static Methods

OracleRefCursor static methods are listed in [Table 13-28](#).

Table 13-28 OracleRefCursor Static Methods

Methods	Description
Equals	Inherited from System.Object (Overloaded)

OracleRefCursor Static Fields

OracleRefCursor static field is listed in [Table 13-29](#).

Table 13-29 OracleRefCursor Static Field

Methods	Description
Null	Represents a null value that can be assigned to an OracleRefCursor instance

OracleRefCursor Properties

OracleRefCursor properties are listed in [Table 13-30](#).

Table 13-30 OracleRefCursor Properties

Properties	Description
Connection	A reference to the OracleConnection used to fetch the REF CURSOR data
FetchSize	Specifies the size that the OracleDataReader internal cache needs to store result set data
IsNull	Indicates whether or not the OracleRefCursor is null
RowSize	Specifies the amount of memory the OracleRefcursor internal cache needs to store one row of data

Table 13-30 (Cont.) OracleRefCursor Properties

Properties	Description
RowsToFetchPerRoundTrip	Specifies the total number of rows to retrieve per database round trip <i>Not Available in ODP.NET, Unmanaged Driver</i>

OracleRefCursor Instance Methods

OracleRefCursor instance methods are listed in [Table 13-31](#).

Table 13-31 OracleRefCursor Instance Methods

Methods	Description
Dispose	Disposes the resources allocated by the OracleRefCursor object
Equals	Inherited from System.Object (Overloaded)
GetDataReader	Returns an OracleDataReader object for the REF CURSOR
GetHashCode	Inherited from System.Object
GetType	Inherited from System.Object
ToString	Inherited from System.Object

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleRefCursor Class](#)

OracleRefCursor Static Methods

OracleRefCursor static methods are listed in [Table 13-32](#).

Table 13-32 OracleRefCursor Static Methods

Methods	Description
Equals	Inherited from System.Object (Overloaded)

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRefCursor Class](#)
- [OracleRefCursor Members](#)

OracleRefCursor Static Fields

OracleRefCursor static field is listed in [Table 13-32](#).

Table 13-33 OracleRefCursor Static Field

Methods	Description
Null	Represents a null value that can be assigned to an OracleRefCursor instance

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRefCursor Class](#)
- [OracleRefCursor Members](#)

Null

This static field represents a null value that can be assigned to an OracleRefCursor instance.

Declaration

```
// C#
public static readonly OracleRefCursor Null;
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRefCursor Class](#)
- [OracleRefCursor Members](#)

OracleRefCursor Properties

OracleRefCursor properties are listed in [Table 13-34](#).

Table 13-34 OracleRefCursor Properties

Properties	Description
Connection	A reference to the OracleConnection used to fetch the REF CURSOR data
FetchSize	Specifies the size that the OracleDataReader internal cache needs to store result set data
IsNull	Indicates whether or not the OracleRefCursor is null
RowSize	Specifies the amount of memory the OracleRefcursor internal cache needs to store one row of data
RowsToFetchPerRoundTrip	Specifies the total number of rows to retrieve per database round trip <i>Not Available in ODP.NET, Unmanaged Driver</i>

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleRefCursor Class](#)
- [OracleRefCursor Members](#)

Connection

This property refers to the OracleConnection used to fetch the REF CURSOR data.

Declaration

```
// C#  
public OracleConnection Connection {get;}
```

Property Value

An OracleConnection.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

This property is bound to a REF CURSOR once it is set. After the OracleRefCursor object is created by the constructor, this property is initially null. An OracleRefCursor object can be bound to a REF CURSOR after a command execution.

If the connection is closed or returned to the connection pool, the OracleRefCursor is placed in an uninitialized state and no operation can be carried out from it. However, the uninitialized OracleRefCursor can be reassigned to another REF CURSOR.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRefCursor Class](#)
- [OracleRefCursor Members](#)

FetchSize

This property specifies the size that the `OracleDataReader` internal cache needs to store result set data.

Declaration

```
// C#  
public long FetchSize {get; set;}
```

Property Value

A long that specifies the size (in bytes) of the `OracleRefCursor` internal cache.

Exceptions

`ArgumentException` - The `FetchSize` value specified is invalid.

Remarks

Default = 131072.

The `FetchSize` property value is inherited by the `OracleCommand` that created the `OracleRefCursor` object. The `FetchSize` property on the `OracleDataReader` object determines the amount of data the `OracleRefCursor` fetches into its internal cache for each database round-trip.

This property is useful if the `OracleRefCursor` is explicitly used to fill the `DataSet` or `DataTable` through the `OracleDataAdapter`, because it can provide control on how the data of the `REF CURSOR` is fetched.

If an `OracleDataReader` object is created from the `OracleRefCursor`, the resulting `OracleDataReader` object inherits the `FetchSize` value of the `OracleDataReader` object. However, the inherited value can be overridden, if it is set before the first invocation of the `OracleDataReader` `Read` method for the given result set, by setting the `OracleDataReader` `FetchSize` property.

The `RowSize` and `FetchSize` properties handle UDT and XMLType data differently than other scalar data types. Because only a reference to the UDT and XMLType data is stored in the ODP.NET's internal cache, the `RowSize` property accounts for only the memory needed for the reference (which is very small) and not the actual size of the UDT and XMLType data. Thus, applications can inadvertently fetch a large number of UDT or XMLType instances from the database in a single database round-trip. This is because the actual size of UDT and XMLType data does not count against the `FetchSize`, and it would require numerous UDT and XMLType references to fill up the default cache size of 131072 bytes. Therefore, when fetching UDT or XMLType data, the `FetchSize` property must be appropriately configured to control the number of UDT and XMLType instances that are to be fetched, rather than the amount of the actual UDT and XMLType data to be fetched.

NOTE: For LOB and LONG data types, only the sizes specified in the `InitialLOBFetchSize` and `InitialLONGFetchSize` properties are accounted for by the `RowSize` property in addition to the metadata and reference information that is maintained by the cache for each LOB in the select list.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRefCursor Class](#)
- [OracleRefCursor Members](#)

IsNull

This property indicates whether or not the `OracleRefCursor` is null.

Declaration

```
// C#  
public bool IsNull {get;}
```

Property Value

Returns true if the `OracleRefCursor` represents a null value. Returns false otherwise.

Exception

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRefCursor Class](#)
- [OracleRefCursor Members](#)

RowSize

This property specifies the amount of memory the `OracleRefCursor` internal cache needs to store one row of data.

Declaration

```
// C#  
public long RowSize {get;}
```

Property Value

A long that indicates the amount of memory (in bytes) that an OracleRefcursor needs to store one row of data for the executed query.

Remarks

The RowSize property is set to a nonzero value when the OracleRefcursor object is created. This property can be used at design time or dynamically during run time, to set the FetchSize, based on number of rows. For example, to enable the OracleRefcursor to fetch *N* rows for each database round-trip, the OracleRefcursor FetchSize property can be set dynamically to RowSize * *N*. Note that for the FetchSize to take effect appropriately, it must be set before the it is used to fill the DataSet/DataTable using OracleDataAdapter.

If an OracleDataReader is obtained from the OracleRefCursor through the GetDataReader method, the resulting OracleDataReader will have its FetchSize property set to the FetchSize value of the OracleRefCursor.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRefCursor Class](#)
- [OracleRefCursor Members](#)

RowsToFetchPerRoundTrip

This property specifies the total number of rows to retrieve per database round trip.

Declaration

```
// C#  
public Int64 RowsToFetchPerRoundTrip { get; set; }
```

Property Type

System.Int64

Remarks

This property has no default value. If it is not set, then ODP.NET ignores this property.

The row data fetched in a single round trip applies to scalar types only, such as NUMBER and VARCHAR2 columns. If reference data types, such as LOBs, UDTs, and XMLType, exist in the result set, they are retrieved in separate round trips. These reference types can have their own fetch tuning parameters as reference type data sizes can vary in size from row to row, sometimes significantly.

RowsToFetchPerRoundTrip can be set before or after the OracleCommand executes on OracleConnection, OracleCommand, OracleDataReader, or OracleRefCursor. The value can also be changed after initial data fetches so that more or fewer rows are fetched on subsequent round trips. For OracleConfiguration, this property can only be set before the first connection opens.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRefCursor Class](#)
- [OracleRefCursor Members](#)

OracleRefCursor Instance Methods

OracleRefCursor instance methods are listed in [Table 13-35](#).

Table 13-35 OracleRefCursor Instance Methods

Methods	Description
Dispose	Disposes the resources allocated by the OracleRefCursor object
Equals	Inherited from System.Object (Overloaded)
GetDataReader	Returns an OracleDataReader object for the REF CURSOR
GetHashCode	Inherited from System.Object
GetType	Inherited from System.Object
ToString	Inherited from System.Object

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRefCursor Class](#)
- [OracleRefCursor Members](#)

Dispose

This instance method disposes of the resources allocated by the OracleRefCursor object.

Declaration

```
// C#  
public void Dispose();
```

Implements

IDisposable

Remarks

The object cannot be reused after being disposed.

Once Dispose() is called, the object of OracleRefCursor is in an uninitialized state. Although some properties can still be accessed, their values may not be accountable. Since resources are freed, method calls can lead to exceptions.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRefCursor Class](#)
- [OracleRefCursor Members](#)

GetDataReader

This instance method returns an `OracleDataReader` object for the REF CURSOR.

Declaration

```
// C#  
public OracleDataReader GetDataReader();
```

Return Value

`OracleDataReader`

Remarks

Using the `OracleDataReader`, rows can be fetched from the REF CURSOR.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRefCursor Class](#)
- [OracleRefCursor Members](#)

Oracle Data Provider for .NET Types Structures

This chapter describes the ODP.NET Types structures.

This chapter contains these topics:

- [OracleBinary Structure](#)
- [OracleBoolean Structure](#)
- [OracleDate Structure](#)
- [OracleDecimal Structure](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalYM Structure](#)
- [OracleString Structure](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampTZ Structure](#)
- [INullable Interface](#)

OracleBinary Structure

The OracleBinary structure represents a variable-length stream of binary data to be stored in or retrieved from a database.

Class Inheritance

System.Object
 System.ValueType
 Oracle.DataAccess.Types.OracleBinary

Declaration

```
// C#  
public struct OracleBinary : IComparable, INullable, IXmlSerializable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#

using System;
using Oracle.DataAccess.Types;

class OracleBinarySample
{
    static void Main(string[] args)
    {
        // Initialize the OracleBinary structures
        OracleBinary binary1= new OracleBinary(new byte[] {1,2,3,4,5});
        OracleBinary binary2 = new OracleBinary(new byte[] {1,2,3});
        OracleBinary binary3 = new OracleBinary(new byte[] {4,5});
        OracleBinary binary4 = binary2 + binary3;

        // Compare binary1 and binary4; they're equal
        if (binary1 == binary4)
            Console.WriteLine("The two OracleBinary structs are equal");
        else
            Console.WriteLine("The two OracleBinary structs are different");
    }
}
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Members](#)
- [OracleBinary Constructor](#)
- [OracleBinary Static Fields](#)
- [OracleBinary Static Methods](#)
- [OracleBinary Static Operators](#)
- [OracleBinary Static Type Conversion Operators](#)
- [OracleBinary Properties](#)
- [OracleBinary Instance Methods](#)

OracleBinary Members

OracleBinary members are listed in the following tables:

OracleBinary Constructors

OracleBinary constructors are listed in [Table 14-1](#)

Table 14-1 OracleBinary Constructors

Constructor	Description
OracleBinary Constructor	Instantiates a new instance of OracleBinary structure

OracleBinary Static Fields

The OracleBinary static fields are listed in [Table 14-2](#).

Table 14-2 OracleBinary Static Fields

Field	Description
Null	Represents a null value that can be assigned to an instance of the OracleBinary structure

OracleBinary Static Methods

The OracleBinary static methods are listed in [Table 14-3](#).

Table 14-3 OracleBinary Static Methods

Methods	Description
Concat	Returns the concatenation of two OracleBinary structures
Equals	Determines if two OracleBinary values are equal (Overloaded)
GetXsdType	Returns the XML Schema definition language (XSD) of the specified XmlSchemaSet
GreaterThan	Determines if the first of two OracleBinary values is greater than the second
GreaterThanOrEqual	Determines if the first of two OracleBinary values is greater than or equal to the second
LessThan	Determines if the first of two OracleBinary values is less than the second
LessThanOrEqual	Determines if the first of two OracleBinary values is less than or equal to the second
NotEquals	Determines if two OracleBinary values are not equal

OracleBinary Static Operators

The OracleBinary static operators are listed in [Table 14-4](#).

Table 14-4 OracleBinary Static Operators

Operator	Description
operator +	Concatenates two OracleBinary values
operator ==	Determines if two OracleBinary values are equal
operator >	Determines if the first of two OracleBinary values is greater than the second
operator >=	Determines if the first of two OracleBinary values is greater than or equal to the second
operator !=	Determines if two OracleBinary values are not equal
operator <	Determines if the first of two OracleBinary value is less than the second
operator <=	Determines if the first of two OracleBinary value is less than or equal to the second

OracleBinary Static Type Conversion Operators

The OracleBinary static type conversion operators are listed in [Table 14-5](#).

Table 14-5 OracleBinary Static Type Conversion Operators

Operator	Description
explicit operator byte[]	Converts an instance value to a byte array
implicit operator OracleBinary	Converts an instance value to an OracleBinary structure

OracleBinary Properties

The OracleBinary properties are listed in [Table 14-6](#).

Table 14-6 OracleBinary Properties

Properties	Description
IsNull	Indicates whether or not the current instance has a null value
Item	Obtains the particular byte in an OracleBinary structure using an index
Length	Returns the length of the binary data
Value	Returns the binary data that is stored in an OracleBinary structure

OracleBinary Instance Methods

The OracleBinary instance methods are listed in [Table 14-7](#).

Table 14-7 OracleBinary Instance Methods

Methods	Description
CompareTo	Compares the current instance to an object and returns an integer that represents their relative values

Table 14-7 (Cont.) OracleBinary Instance Methods

Methods	Description
Equals	Determines if two objects contain the same binary data (Overloaded)
GetHashCode	Returns a hash code for the current instance
GetType	Inherited from System.Object
ToString	Converts the current OracleBinary structure to a string

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)

OracleBinary Constructor

The OracleBinary constructor instantiates a new instance of the OracleBinary structure and sets its value to the provided array of bytes.

Declaration

```
// C#  
public OracleBinary(byte[ ] bytes);
```

Parameters

- *bytes*
A byte array.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

OracleBinary Static Fields

The OracleBinary static fields are listed in [Table 14-8](#).

Table 14-8 OracleBinary Static Fields

Field	Description
Null	Represents a null value that can be assigned to an instance of the OracleBinary structure

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

Null

This static field represents a null value that can be assigned to an instance of the OracleBinary structure.

Declaration

```
// C#
public static readonly OracleBinary Null;
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

OracleBinary Static Methods

The OracleBinary static methods are listed in [Table 14-9](#).

Table 14-9 OracleBinary Static Methods

Methods	Description
Concat	Returns the concatenation of two OracleBinary structures
Equals	Determines if two OracleBinary values are equal (Overloaded)
GetXsdType	Returns the XML Schema definition language (XSD) of the specified XmlSchemaSet
GreaterThan	Determines if the first of two OracleBinary values is greater than the second
GreaterThanOrEqual	Determines if the first of two OracleBinary values is greater than or equal to the second
LessThan	Determines if the first of two OracleBinary values is less than the second
LessThanOrEqual	Determines if the first of two OracleBinary values is less than or equal to the second
NotEquals	Determines if two OracleBinary values are not equal

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

Concat

This method returns the concatenation of two OracleBinary structures.

Declaration

```
// C#  
public static OracleBinary Concat(OracleBinary value1, OracleBinary value2);
```

Parameters

- *value1*
The first OracleBinary.
- *value2*
The second OracleBinary.

Return Value

An OracleBinary.

Remarks

If either argument has a null value, the returned OracleBinary structure has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

Equals

This method determines if two OracleBinary values are equal.

Declaration

```
// C#  
public static bool Equals(OracleBinary value1, OracleBinary value2);
```

Parameters

- *value1*

The first `OracleBinary`.

- *value2*

The second `OracleBinary`.

Return Value

Returns true if two `OracleBinary` values are equal; otherwise returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleBinary` that has a value is greater than an `OracleBinary` that has a null value.
- Two `OracleBinary`s that contain a null value are equal.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

GetXsdType

This method returns the XML Schema definition language (XSD) of the specified `XmlSchemaSet`.

Declaration

```
// C#  
public static XmlQualifiedName GetXsdType(XmlSchemaSet schemaSet);
```

Parameters

- *schemaSet*
An `XmlSchemaSet`.

Return Value

Returns a string that indicates the XSD of the specified `XmlSchemaSet`.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

GreaterThan

This method determines whether or not the first of two `OracleBinary` values is greater than the second.

Declaration

```
// C#  
public static bool GreaterThan(OracleBinary value1, OracleBinary value2);
```

Parameters

- *value1*
The first `OracleBinary`.
- *value2*
The second `OracleBinary`.

Return Value

Returns true if the first of two `OracleBinary` values is greater than the second; otherwise returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleBinary` that has a value is greater than an `OracleBinary` that has a null value.
- Two `OracleBinary`s that contain a null value are equal.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Types;  
  
class GreaterThanSample  
{  
    static void Main(string[] args)  
    {  
        OracleBinary binary1 = OracleBinary.Null;  
        OracleBinary binary2 = new OracleBinary(new byte[] {1});  
  
        // Compare two OracleBinary structs; binary1 < binary2  
        if (OracleBinary.GreaterThan(binary1, binary2))  
            Console.WriteLine("binary1 > binary2");  
        else  
            Console.WriteLine("binary1 < binary2");  
    }  
}
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

GreaterThanOrEqual

This method determines whether or not the first of two `OracleBinary` values is greater than or equal to the second.

Declaration

```
// C#  
public static bool GreaterThanOrEqual(OracleBinary value1, OracleBinary value2);
```

Parameters

- *value1*
The first `OracleBinary`.
- *value2*
The second `OracleBinary`.

Return Value

Returns `true` if the first of two `OracleBinary` values is greater than or equal to the second; otherwise returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleBinary` that has a value is greater than an `OracleBinary` that has a null value.
- Two `OracleBinary`s that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

LessThan

This method determines whether or not the first of two `OracleBinary` values is less than the second.

Declaration

```
// C#  
public static bool LessThan(OracleBinary value1, OracleBinary value2);
```

Parameters

- *value1*
The first OracleBinary.
- *value2*
The second OracleBinary.

Return Value

Returns true if the first of two OracleBinary values is less than the second; otherwise returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleBinary that has a value is greater than an OracleBinary that has a null value.
- Two OracleBinaries that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

LessThanOrEqual

This method determines whether or not the first of two OracleBinary values is less than or equal to the second.

Declaration

```
// C#  
public static bool LessThanOrEqual(OracleBinary value1, OracleBinary value2);
```

Parameters

- *value1*
The first OracleBinary.
- *value2*
The second OracleBinary.

Return Value

Returns true if the first of two `OracleBinary` values is less than or equal to the second; otherwise returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleBinary` that has a value is greater than an `OracleBinary` that has a null value.
- Two `OracleBinary`s that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

NotEquals

This method determines whether or not two `OracleBinary` values are not equal.

Declaration

```
// C#  
public static bool NotEquals(OracleBinary value1, OracleBinary value2);
```

Parameters

- *value1*
The first `OracleBinary`.
- *value2*
The second `OracleBinary`.

Return Value

Returns true if two `OracleBinary` values are not equal; otherwise returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleBinary` that has a value is greater than an `OracleBinary` that has a null value.
- Two `OracleBinary`s that contain a null value are equal.

📘 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

OracleBinary Static Operators

The OracleBinary static operators are listed in [Table 14-10](#).

Table 14-10 OracleBinary Static Operators

Operator	Description
operator +	Concatenates two OracleBinary values
operator ==	Determines if two OracleBinary values are equal
operator >	Determines if the first of two OracleBinary values is greater than the second
operator >=	Determines if the first of two OracleBinary values is greater than or equal to the second
operator !=	Determines if two OracleBinary values are not equal
operator <	Determines if the first of two OracleBinary value is less than the second
operator <=	Determines if the first of two OracleBinary value is less than or equal to the second

📘 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

operator +

This method concatenates two OracleBinary values.

Declaration

```
// C#
public static OracleBinary operator + (OracleBinary value1, OracleBinary value2);
```

Parameters

- *value1*
The first OracleBinary.

- *value2*
The second OracleBinary.

Return Value

OracleBinary

Remarks

If either argument has a null value, the returned OracleBinary structure has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

operator ==

This method determines if two OracleBinary values are equal.

Declaration

```
// C#  
public static bool operator == (OracleBinary value1, OracleBinary value2);
```

Parameters

- *value1*
The first OracleBinary.
- *value2*
The second OracleBinary.

Return Value

Returns true if they are the same; otherwise returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleBinary that has a value is greater than an OracleBinary that has a null value.
- Two OracleBinaries that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

operator >

This method determines if the first of two `OracleBinary` values is greater than the second.

Declaration

```
// C#  
public static bool operator > (OracleBinary value1, OracleBinary value2);
```

Parameters

- *value1*
The first `OracleBinary`.
- *value2*
The second `OracleBinary`.

Return Value

Returns `true` if the first of two `OracleBinary` values is greater than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleBinary` that has a value is greater than an `OracleBinary` that has a null value.
- Two `OracleBinary`s that contain a null value are equal.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Types;  
  
class OperatorSample  
{  
    static void Main(string[] args)  
    {  
        OracleBinary binary1 = OracleBinary.Null;  
        OracleBinary binary2 = new OracleBinary(new byte[] {1});  
  
        // Compare two OracleBinary structs; binary1 < binary2  
        if (binary1 > binary2)  
            Console.WriteLine("binary1 > binary2");  
        else  
            Console.WriteLine("binary1 < binary2");  
    }  
}
```

```
}  
}
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

operator >=

This method determines if the first of two `OracleBinary` values is greater than or equal to the second.

Declaration

```
// C#  
public static bool operator >= (OracleBinary value1, OracleBinary value2);
```

Parameters

- *value1*
The first `OracleBinary`.
- *value2*
The second `OracleBinary`.

Return Value

Returns `true` if the first of two `OracleBinary` values is greater than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleBinary` that has a value is greater than an `OracleBinary` that has a null value.
- Two `OracleBinary`s that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

operator !=

This method determines if two `OracleBinary` values are not equal.

Declaration

```
// C#  
public static bool operator != (OracleBinary value1, OracleBinary value2);
```

Parameters

- *value1*
The first OracleBinary.
- *value2*
The second OracleBinary.

Return Value

Returns true if the two OracleBinary values are not equal; otherwise, returns false.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

operator <

This method determines if the first of two OracleBinary values is less than the second.

Declaration

```
// C#  
public static bool operator < ( OracleBinary value1, OracleBinary value2);
```

Parameters

- *value1*
The first OracleBinary.
- *value2*
The second OracleBinary.

Return Value

Returns true if the first of two OracleBinary values is less than the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleBinary that has a value is greater than an OracleBinary that has a null value.
- Two OracleBinaries that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

operator <=

This method determines if the first of two `OracleBinary` values is less than or equal to the second.

Declaration

```
// C#  
public static bool operator <= (OracleBinary value1, OracleBinary value1);
```

Parameters

- *value1*
The first `OracleBinary`.
- *value2*
The second `OracleBinary`.

Return Value

Returns `true` if the first of two `OracleBinary` values is less than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleBinary` that has a value is greater than an `OracleBinary` that has a null value.
- Two `OracleBinary`s that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

OracleBinary Static Type Conversion Operators

The `OracleBinary` static type conversion operators are listed in [Table 14-11](#).

Table 14-11 OracleBinary Static Type Conversion Operators

Operator	Description
explicit operator byte[]	Converts an instance value to a byte array
implicit operator OracleBinary	Converts an instance value to an OracleBinary structure

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

explicit operator byte[]

This method converts an OracleBinary value to a byte array.

Declaration

```
// C#
public static explicit operator byte[] (OracleBinary val);
```

Parameters

- *val*
An OracleBinary.

Return Value

A byte array.

Exceptions

OracleNullValueException - The OracleBinary structure has a null value.

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

implicit operator OracleBinary

This method converts a byte array to an OracleBinary structure.

Declaration

```
// C#
public static implicit operator OracleBinary(byte[ ] bytes);
```

Parameters

- *bytes*
A byte array.

Return Value

OracleBinary

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

OracleBinary Properties

The OracleBinary properties are listed in [Table 14-12](#).

Table 14-12 OracleBinary Properties

Properties	Description
IsNull	Indicates whether or not the current instance has a null value
Item	Obtains the particular byte in an OracleBinary structure using an index
Length	Returns the length of the binary data
Value	Returns the binary data that is stored in an OracleBinary structure

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

IsNull

This property indicates whether or not the current instance has a null value.

Declaration

```
// C#  
public bool IsNull {get;}
```

Property Value

Returns true if the current instance has a null value; otherwise returns false.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

Item

This property obtains the particular byte in an OracleBinary structure using an index.

Declaration

```
// C#  
public byte this[int index] {get;}
```

Property Value

A byte in the specified index.

Exceptions

OracleNullValueException - The current instance has a null value.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Types;  
  
class ItemSample  
{  
    static void Main(string[] args)  
    {  
        OracleBinary binary = new OracleBinary(new byte[] {1,2,3,4});  
  
        // Prints the value 4  
        Console.WriteLine(binary[binary.Length - 1]);  
    }  
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

Length

This property returns the length of the binary data.

Declaration

```
// C#  
public int length {get;}
```

Property Value

Length of the binary data.

Exceptions

OracleNullValueException - The current instance has a null value.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Types;  
  
class LengthSample  
{  
    static void Main(string[] args)  
    {  
        OracleBinary binary = new OracleBinary(new byte[] {1,2,3,4});  
  
        // Prints the value 4  
        Console.WriteLine(binary.Length);  
    }  
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

Value

This property returns the binary data that is stored in the OracleBinary structure.

Declaration

```
// C#  
public byte[] Value {get;}
```

Property Value

Binary data.

Exceptions

OracleNullValueException - The current instance has a null value.

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

OracleBinary Instance Methods

The OracleBinary instance methods are listed in [Table 14-13](#).

Table 14-13 OracleBinary Instance Methods

Methods	Description
CompareTo	Compares the current instance to an object and returns an integer that represents their relative values
Equals	Determines if two objects contain the same binary data (Overloaded)
GetHashCode	Returns a hash code for the current instance
GetType	Inherited from System.Object
ToString	Converts the current OracleBinary structure to a string

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

CompareTo

This method compares the current instance to an object and returns an integer that represents their relative values

Declaration

```
// C#  
public int CompareTo(object obj);
```

Parameters

- *obj*
The object being compared.

Return Value

The method returns a number that is:

- Less than zero: if the current OracleBinary instance value is less than *obj*.
- Zero: if the current OracleBinary instance and *obj* values have the same binary data.
- Greater than zero: if the current OracleBinary instance value is greater than *obj*.

Implements

Comparable

Exceptions

ArgumentException - The parameter is not of type OracleBinary.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between OracleBinarys. For example, comparing an OracleBinary instance with an OracleTimeStamp instance is not allowed. When an OracleBinary is compared with a different type, an ArgumentException is thrown.
- Any OracleBinary that has a value is greater than an OracleBinary that has a null value.
- Two OracleBinarys that contain a null value are equal.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Types;  
  
class CompareToSample  
{  
    static void Main(string[] args)  
    {  
        OracleBinary binary1 = new OracleBinary(new byte[] {1,2,3});  
        OracleBinary binary2 = new OracleBinary(new byte[] {1,2,3,4});  
  
        // Compare  
        if (binary1.CompareTo(binary2) == 0)  
            Console.WriteLine("binary1 is the same as binary2");  
        else  
            Console.WriteLine("binary1 is different from binary2");  
    }  
}
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

Equals

This method determines whether or not an object is an instance of `OracleBinary`, and has the same binary data as the current instance.

Declaration

```
// C#  
public override bool Equals(object obj);
```

Parameters

- *obj*
The object being compared.

Return Value

Returns `true` if *obj* is an instance of `OracleBinary`, and has the same binary data as the current instance; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleBinary` that has a value is greater than an `OracleBinary` that has a null value.
- Two `OracleBinary`s that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

GetHashCode

Overrides `Object`

This method returns a hash code for the `OracleBinary` instance.

Declaration

```
// C#  
public override int GetHashCode();
```

Return Value

An int that represents the hash.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

ToString

Overrides Object

This method converts an OracleBinary instance to a string instance.

Declaration

```
// C#  
public override string ToString();
```

Return Value

string

Remarks

If the current OracleBinary instance has a null value, the returned string "null".

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBinary Structure](#)
- [OracleBinary Members](#)

OracleBoolean Structure

The OracleBoolean structure represents a logical value that is either TRUE or FALSE.

ODP.NET, Unmanaged Driver can access Oracle Database PL/SQL Booleans in Oracle Database Release 12.1 and later. ODP.NET Core and ODP.NET, Managed Driver can access Oracle Database PL/SQL Booleans in Oracle Database Release 12.2 and later. Starting with version 23, all ODP.NET providers support Boolean table columns introduced in Oracle AI Database 26ai.

OracleBoolean can use the following string literals as TRUE or FALSE values. These values are accepted in a case insensitive manner.

Table 14-14 Accepted OracleBoolean Values

TRUE values	False values
true	false
t	f
yes	no
y	n
on	off
1	0

If whitespace is present before or after the string literals, then the whitespace is not considered upon .NET Boolean conversion.

If the string literals are anything other than those listed above, then an `InvalidCastException` will be returned when trying to read the value.

Class Inheritance

System.Object
 System.ValueType
 Oracle.DataAccess.Types.OracleBoolean

Declaration

```
// C#  
public struct OracleBoolean : IComparable, INullable, IXmlSerializable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

A `OracleBoolean` structure represents three possible values: `TRUE`, `FALSE`, and `NULL`. A non-zero value is interpreted as `TRUE`.

Example

```
// C#
using System;
using System.Data;
using Oracle.DataAccess.Types; // for use with ODP.NET, Unmanaged Driver
// using Oracle.ManagedDataAccess.Types; // for use with ODP.NET, Managed Driver

class OracleBooleanSample
{
    static void Main(string[] args)
    {
        OracleBoolean oracleBoolean1 = new OracleBoolean(true);
        OracleBoolean oracleBoolean2 = new OracleBoolean(0);

        Console.WriteLine("oracleBoolean1 : " + oracleBoolean1);
        Console.WriteLine("oracleBoolean2 : " + oracleBoolean2);
    }
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Members](#)
- [OracleBoolean Constructors](#)
- [OracleBoolean Static Fields](#)
- [OracleBoolean Static Methods](#)
- [OracleBoolean Static Operators](#)
- [OracleBoolean Static Type Conversions](#)
- [OracleBoolean Properties](#)
- [OracleBoolean Instance Methods](#)

OracleBoolean Members

OracleBoolean members are listed in the following tables:

OracleBoolean Constructors

OracleBoolean constructors are listed in [Table 14-15](#)

Table 14-15 OracleBoolean Constructors

Constructor	Description
OracleBoolean Constructors	Instantiates a new instance of OracleBoolean structure (Overloaded)

OracleBoolean Static Fields

The OracleBoolean static fields are listed in [Table 14-16](#).

Table 14-16 OracleBoolean Static Fields

Field	Description
False	Represents a false value that can be assigned to an OracleBoolean instance
Null	Represents a null value that can be assigned to an OracleBoolean instance
One	Indicates a constant representing the positive one value
True	Represents a true value that can be assigned to an OracleBoolean instance
Zero	Indicates a constant representing the zero value

OracleBoolean Static Methods

OracleBoolean static methods are listed in [Table 14-17](#)

Table 14-17 OracleBoolean Static Methods

Methods	Description
And	Returns the result of bitwise AND operation of two OracleBoolean instances
Equals	Determines whether or not the two OracleBoolean values are equal
GreaterThan	Determines whether or not the first of two OracleBoolean values is greater than the second
GreaterThanOrEquals	Determines whether or not the first of two OracleBoolean values is greater than or equal to the second
LessThan	Determines whether or not the first of two OracleBoolean values is less than the second
LessThanOrEquals	Determines whether or not the first of two OracleBoolean values is less than or equal to the second
NotEquals	Determines whether or not two OracleBoolean values are not equal
OnesComplement	Returns the result of a one's complement operation on the specified OracleBoolean value
Or	Returns the result of bitwise OR operation of two OracleBoolean instances
Parse	Returns an OracleBoolean structure and sets its value using a string
Xor	Returns the result of a bitwise exclusive OR operation of two OracleBoolean instances

OracleBoolean Static Operators

The OracleBoolean static operators are listed in [Table 14-18](#).

Table 14-18 OracleBoolean Static Operators

Field	Description
operator >	Determines whether or not the first of two OracleBoolean values is greater than the second
operator >=	Determines whether or not the first of two OracleBoolean values is greater than or equal to the second
operator <	Determines whether or not the first of two OracleBoolean values is less than the second
operator <=	Determines whether or not the first of two OracleBoolean values is less than or equal to the second
operator ==	Indicates whether or not the two OracleBoolean instances are equal
operator !=	Determines whether or not two OracleBoolean values are not equal
operator !	Determines the result of a NOT operation on a OracleBoolean
operator ~	Returns the result of a one's complement operation on the specified OracleBoolean value
operator false	Determines whether or not the specified OracleBoolean value is false
operator true	Determines whether or not the specified OracleBoolean value is true
operator &	Returns the result of bitwise AND operation of two OracleBoolean instances
operator 	Returns the result of bitwise OR operation of two OracleBoolean instances
operator ^	Returns the result of bitwise exclusive OR operation of two OracleBoolean instances

The OracleBoolean Static Type conversions

The OracleBoolean static type conversions are listed in [Table 14-19](#)

Table 14-19 OracleBoolean Static Type Conversions

Field	Description
implicit operator OracleBoolean	Returns the OracleBoolean representation of a boolean value
explicit operator bool	Returns the boolean representation of the OracleBoolean value
explicit operator OracleBoolean	Converts a structure to an OracleBoolean structure (Overloaded)

OracleBoolean Properties

The OracleBoolean properties are listed in [Table 14-26](#).

Table 14-20 OracleBoolean Properties

Properties	Description
ByteValue	Returns a byte that represents the OracleBoolean structure
IsFalse	Indicates whether or not the value of the current instance is false

Table 14-20 (Cont.) OracleBoolean Properties

Properties	Description
IsNull	Indicates whether or not the current instance has a null value
IsTrue	Indicates whether or not the value of the current instance is true
Value	Returns a boolean value that represents the current instance

OracleBoolean Instance Methods

The OracleBoolean instance methods are listed in [Table 14-21](#).

Table 14-21 OracleBoolean Instance Methods

Method	Description
CompareTo	Compares the current instance to the supplied object and returns an integer that represents their relative values
Equals	Determines whether or not an object is an instance of OracleBoolean, and whether or not the value of the object is equal to the current instance
GetHashCode	Returns a hash code for the current instance
ToString	Returns the string representation of the current instance

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleBoolean Structure](#)

OracleBoolean Constructors

The OracleBoolean constructors instantiates a new instance of the OracleBoolean structure.

Overload List:

- [OracleBoolean\(bool\)](#)

This constructor creates a new instance of the OracleBoolean structure and sets its value to the supplied Boolean value.

- [OracleBoolean\(int\)](#)

This constructor creates a new instance of the OracleBoolean structure and sets its value to the supplied Int32 value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

OracleBoolean(bool)

This constructor creates a new instance of the `OracleBoolean` structure and sets its value to the supplied `Boolean` value.

Declaration

```
// C#  
public OracleBoolean(bool value) ;
```

Parameters

- *value*
The provided `Boolean` value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

OracleBoolean(int)

This constructor creates a new instance of the `OracleBoolean` structure and sets its value to the supplied `Int32` value.

Declaration

```
// C#  
public OracleBoolean(int value)
```

Parameters

- *value*
The provided `Int32` value.

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

OracleBoolean Static Fields

The OracleBoolean static fields are listed in [Table 14-22](#).

Table 14-22 OracleBoolean Static Fields

Field	Description
False	Represents a false value that can be assigned to an OracleBoolean instance
Null	Represents a null value that can be assigned to an OracleBoolean instance
One	Indicates a constant representing the positive one value
True	Represents a true value that can be assigned to an OracleBoolean instance
Zero	Indicates a constant representing the zero value

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

False

This static field represents a false value that can be assigned to an OracleBoolean instance.

Declaration

```
// C#  
public static readonly OracleBoolean False;
```

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

Null

This static field represents a null value that can be assigned to an OracleBoolean instance.

Declaration

```
// C#  
public static readonly OracleBoolean Null;
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

One

This static field indicates a constant representing the positive one value.

Declaration

```
// C#  
public static readonly OracleBoolean One;
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

True

This static field represents a true value that can be assigned to an OracleBoolean instance.

Declaration

```
// C#  
public static readonly OracleBoolean True;
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

Zero

This static field indicates a constant representing the zero value.

Declaration

```
// C#  
public static readonly OracleBoolean Zero;
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

OracleBoolean Static Methods

OracleBoolean static methods are listed in [Table 14-23](#)

Table 14-23 OracleBoolean Static Methods

Methods	Description
And	Returns the result of bitwise AND operation of two OracleBoolean instances
Equals	Determines whether or not the two OracleBoolean values are equal
GreaterThan	Determines whether or not the first of two OracleBoolean values is greater than the second
GreaterThanOrEquals	Determines whether or not the first of two OracleBoolean values is greater than or equal to the second
LessThan	Determines whether or not the first of two OracleBoolean values is less than the second
LessThanOrEquals	Determines whether or not the first of two OracleBoolean values is less than or equal to the second
NotEquals	Determines whether or not two OracleBoolean values are not equal
OnesComplement	Returns the result of a one's complement operation on the specified OracleBoolean value
Or	Returns the result of bitwise OR operation of two OracleBoolean instances
Parse	Returns an OracleBoolean structure and sets its value using a string
Xor	Returns the result of a bitwise exclusive OR operation of two OracleBoolean instances

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

And

This method returns the result of bitwise AND operation of two OracleBoolean instances.

Declaration

```
// C#  
public static OracleBoolean And(OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *value1*
An OracleBoolean instance
- *value2*
An OracleBoolean instance

Return Value

An OracleBoolean that contains the value of the result of bitwise AND operation of two OracleBoolean instances.

Remarks

If either of the specified OracleBoolean instances is null, an OracleBoolean with a null value will be returned.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

Equals

This method returns an OracleBoolean that indicates whether or not the two OracleBoolean instances are equal.

Declaration

```
// C#  
public static OracleBoolean Equal(OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *value1*
An OracleBoolean instance
- *value2*
An OracleBoolean instance

Return Value

An OracleBoolean that is true if the specified two OracleBoolean instances are equal; otherwise, returns an OracleBoolean that is false.

Remarks

If either of the specified OracleBoolean instances is null, an OracleBoolean with a null value will be returned.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

GreaterThan

This method determines if the first of two OracleBoolean values is greater than the second.

Declaration

```
// C#  
public static OracleBoolean GreaterThan(OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *value1*
The first OracleBoolean
- *value2*
The second OracleBoolean

Return Value

An OracleBoolean that is true if the first of two OracleBoolean values is greater than the second; otherwise, returns false.

Remarks

If either of the specified OracleBoolean instances is null, an OracleBoolean with a null value will be returned.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

GreaterThanOrEquals

This method determines if the first of two `OracleBoolean` values is greater than or equal to the second.

Declaration

```
// C#  
public static OracleBoolean GreaterThanOrEquals(OracleBoolean value1, OracleBoolean  
value2);
```

Parameters

- *value1*
The first `OracleBoolean`
- *value2*
The second `OracleBoolean`

Return Value

An `OracleBoolean` that is true if the first of two `OracleBoolean` values is greater than or equal to the second; otherwise, returns false.

Remarks

If either of the specified `OracleBoolean` instances is null, an `OracleBoolean` with a null value will be returned.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

LessThan

This method determines if the first of two `OracleBoolean` values is less than the second.

Declaration

```
// C#  
public static OracleBoolean LessThan(OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *value1*
The first OracleBoolean
- *value2*
The second OracleBoolean

Return Value

An OracleBoolean that is true if the first of two OracleBoolean values is less than the second; otherwise, returns false.

Remarks

If either of the specified OracleBoolean instances is null, an OracleBoolean with a null value will be returned.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

LessThanOrEquals

This method determines if the first of two OracleBoolean values is less or equal than the second.

Declaration

```
// C#  
public static OracleBoolean LessThanOrEquals(OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *value1*
The first OracleBoolean
- *value2*
The second OracleBoolean

Return Value

An OracleBoolean that is true if the first of two OracleBoolean values is less than or equal to the second; otherwise, returns false.

Remarks

If either of the specified OracleBoolean instances is null, an OracleBoolean with a null value will be returned.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

NotEquals

This method determines if two OracleBoolean values are not equal.

Declaration

```
// C#  
public static OracleBoolean NotEquals(OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *value1*
The first OracleBoolean
- *value2*
The second OracleBoolean

Return Value

An OracleBoolean that is true if two OracleBoolean values are not equal; otherwise, returns false.

Remarks

If either of the specified OracleBoolean instances is null, an OracleBoolean with a null value will be returned.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

OnesComplement

This method returns the result of a one's complement operation on the specified OracleBoolean value.

Declaration

```
// C#  
public static OracleBoolean OnesComplement(OracleBoolean value1);
```

Parameters

- *value1*
An OracleBoolean instance

Return Value

An OracleBoolean that contains the value of the result of a one's complement operation on the specified OracleBoolean value.

Remarks

If the specified OracleBoolean instance is null, an OracleBoolean with a null value will be returned.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

Or

This method returns the result of bitwise OR operation of two OracleBoolean instances.

Declaration

```
// C#  
public static OracleBoolean Or(OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *value1*
An OracleBoolean instance
- *value2*
An OracleBoolean instance

Return Value

An OracleBoolean that contains the value of the result of bitwise OR operation of two OracleBoolean instances.

Remarks

If either of the specified OracleBoolean instances is null, an OracleBoolean with a null value will be returned.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

Parse

This method converts a string to an OracleBoolean.

Declaration

```
// C#  
public static OracleBoolean Parse(string str);
```

Parameters

- *str*
The string being converted.

Return Value

A new OracleBoolean structure.

Exceptions

ArgumentNullException – The *str* parameter is null.

IndexOutOfRangeException – The *str* parameter is an empty string.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

Xor

This method returns the result of a bitwise exclusive OR operation of two OracleBoolean instances.

Declaration

```
// C#  
public static OracleBoolean Xor(OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *value1*
An OracleBoolean instance

- `value2`
An OracleBoolean instance

Return Value

An OracleBoolean that contains the value of the result of bitwise exclusive OR operation of two OracleBoolean instances.

Remarks

If either of the specified OracleBoolean instances is null, an OracleBoolean with a null value will be returned.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

OracleBoolean Static Operators

The OracleBoolean static operators are listed in [Table 14-24](#).

Table 14-24 OracleBoolean Static Operators

Field	Description
operator >	Determines whether or not the first of two OracleBoolean values is greater than the second
operator >=	Determines whether or not the first of two OracleBoolean values is greater than or equal to the second
operator <	Determines whether or not the first of two OracleBoolean values is less than the second
operator <=	Determines whether or not the first of two OracleBoolean values is less than or equal to the second
operator ==	Indicates whether or not the two OracleBoolean instances are equal
operator !=	Determines whether or not two OracleBoolean values are not equal
operator !	Determines the result of a NOT operation on a OracleBoolean
operator ~	Returns the result of a one's complement operation on the specified OracleBoolean value
operator false	Determines whether or not the specified OracleBoolean value is false
operator true	Determines whether or not the specified OracleBoolean value is true
operator &	Returns the result of bitwise AND operation of two OracleBoolean instances
operator 	Returns the result of bitwise OR operation of two OracleBoolean instances

Table 14-24 (Cont.) OracleBoolean Static Operators

Field	Description
operator ^	Returns the result of bitwise exclusive OR operation of two OracleBoolean instances

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

operator >

This method determines whether or not the first of two OracleBoolean values is greater than the second.

Declaration

```
// C#  
public static operator > (OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *value1*
An OracleBoolean instance
- *value2*
An OracleBoolean instance

Return Value

An OracleBoolean that is true if the first of two OracleBoolean values is greater than the second; otherwise, returns false.

Remarks

If either of the specified OracleBoolean instances is null, an OracleBoolean with a null value will be returned.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

operator >=

This method determines whether or not the first of two `OracleBoolean` values is greater than or equal to the second.

Declaration

```
// C#  
public static operator >= (OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *value1*
An `OracleBoolean` instance
- *value2*
An `OracleBoolean` instance

Return Value

An `OracleBoolean` that is true if the first of two `OracleBoolean` values is greater than or equal to the second; otherwise, returns false.

Remarks

If either of the specified `OracleBoolean` instances is null, an `OracleBoolean` with a null value will be returned.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

operator <

This method determines whether or not the first of two `OracleBoolean` values is less than the second.

Declaration

```
// C#  
public static operator < (OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *value1*
An `OracleBoolean` instance
- *value2*
An `OracleBoolean` instance

Return Value

An OracleBoolean that is true if the first of two OracleBoolean values is less than the second; otherwise, returns false.

Remarks

If either of the specified OracleBoolean instances is null, an OracleBoolean with a null value will be returned.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

operator <=

This method determines whether or not the first of two OracleBoolean values is less than or equal to the second.

Declaration

```
// C#  
public static operator <= (OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *value1*
An OracleBoolean instance
- *value2*
An OracleBoolean instance

Return Value

An OracleBoolean that is true if the first of two OracleBoolean values is less than or equal to the second; otherwise, returns false.

Remarks

If either of the specified OracleBoolean instances is null, an OracleBoolean with a null value will be returned.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

operator ==

This method returns an `OracleBoolean` that indicates whether or not the two `OracleBoolean` instances are equal.

Declaration

```
// C#  
public static operator == (OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *value1*
An `OracleBoolean` instance
- *value2*
An `OracleBoolean` instance

Return Value

An `OracleBoolean` that is true if the specified two `OracleBoolean` instances are equal; otherwise, returns false.

Remarks

If either of the specified `OracleBoolean` instances is null, an `OracleBoolean` with a null value will be returned.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

operator !=

This method determines whether or not two `OracleBoolean` values are not equal.

Declaration

```
// C#  
public static operator != (OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *value1*
An `OracleBoolean` instance
- *value2*
An `OracleBoolean` instance

Return Value

An OracleBoolean that is true if two OracleBoolean values are not equal; otherwise, returns false.

Remarks

If either of the specified OracleBoolean instances is null, an OracleBoolean with a null value will be returned.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

operator !

This method determines the result of a NOT operation on a OracleBoolean.

Declaration

```
// C#  
public static operator ! (OracleBoolean value1);
```

Parameters

- *value1*
An OracleBoolean instance

Return Value

An OracleBoolean that is true if the specified OracleBoolean value is true; otherwise, returns false.

Remarks

If the specified OracleBoolean instance is null, an OracleBoolean with a null value will be returned.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

operator ~

This method returns the result of a one's complement operation on the specified OracleBoolean value.

Declaration

```
// C#  
public static operator ~ (OracleBoolean value1);
```

Parameters

- *value1*
An OracleBoolean instance

Return Value

An OracleBoolean that contains the value of the result of a one's complement operation on the specified OracleBoolean value.

Remarks

If the specified OracleBoolean instance is null, an OracleBoolean with a null value will be returned.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

operator false

This method determines whether or not the specified OracleBoolean value is false.

Declaration

```
// C#  
public static operator false (OracleBoolean value1);
```

Parameters

- *value1*
An OracleBoolean instance

Return Value

An OracleBoolean that is true if specified OracleBoolean value is false; otherwise, returns false.

Remarks

This property will return false if the current instance is null.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

operator true

This method determines whether or not the specified OracleBoolean value is true.

Declaration

```
// C#  
public static operator true (OracleBoolean value1);
```

Parameters

- *value1*
An OracleBoolean instance

Return Value

An OracleBoolean that is true if specified OracleBoolean value is true; otherwise, returns false.

Remarks

This property will return false if the current instance is null.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

operator &

This method returns the result of bitwise AND operation of two OracleBoolean instances.

Declaration

```
// C#  
public static operator & (OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *value1*
An OracleBoolean instance
- *value2*

An OracleBoolean instance

Return Value

An OracleBoolean that contains the value of the result of bitwise AND operation of two OracleBoolean instances.

Remarks

If either of the specified OracleBoolean instances is null, an OracleBoolean with a null value will be returned.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

operator |

This method returns the result of bitwise OR operation of two OracleBoolean instances.

Declaration

```
// C#  
public static operator | (OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *value1*
An OracleBoolean instance
- *value2*
An OracleBoolean instance

Return Value

An OracleBoolean that contains the value of the result of bitwise OR operation of two OracleBoolean instances.

Remarks

If either of the specified OracleBoolean instances is null, an OracleBoolean with a null value will be returned.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

operator ^

This method returns the result of bitwise exclusive OR operation of two OracleBoolean instances.

Declaration

```
// C#
public static operator ^ (OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *value1*
An OracleBoolean instance
- *value2*
An OracleBoolean instance

Return Value

An OracleBoolean that contains the value of the result of bitwise exclusive OR operation of two OracleBoolean instances.

Remarks

If either of the specified OracleBoolean instances is null, an OracleBoolean with a null value will be returned.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

OracleBoolean Static Type Conversions

The OracleBoolean static type conversions are listed in [Table 14-25](#)

Table 14-25 OracleBoolean Static Type Conversions

Field	Description
implicit operator OracleBoolean	Returns the OracleBoolean representation of a boolean value
explicit operator bool	Returns the boolean representation of the OracleBoolean value
explicit operator OracleBoolean	Converts a structure to an OracleBoolean structure (Overloaded)

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

implicit operator OracleBoolean

This method returns the OracleBoolean representation of a boolean value.

Declaration

```
// C#  
public static implicit operator OracleBoolean(bool value1);
```

Parameters

- *value1*
An OracleBoolean instance

Return Value

An OracleBoolean.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

explicit operator bool

This method returns the boolean representation of the OracleBoolean value.

Declaration

```
// C#  
public static explicit operator bool(OracleBoolean value1);
```

Parameters

- *value1*
An OracleBoolean structure

Return Value

A boolean

Exception

OracleNullValueException – OracleBoolean has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

explicit operator OracleBoolean

explicit operator OracleBoolean converts the provided structure to an OracleBoolean structure.

Overload List

- [explicit operator OracleBoolean\(byte\)](#)
This method converts the supplied byte to an OracleBoolean structure.
- [explicit operator OracleBoolean\(Decimal\)](#)
This method converts the supplied Decimal to an OracleBoolean structure.
- [explicit operator OracleBoolean\(Double\)](#)
This method converts the supplied Double to an OracleBoolean structure.
- [explicit operator OracleBoolean\(Int16\)](#)
This method converts the supplied Int16 to an OracleBoolean structure.
- [explicit operator OracleBoolean\(int\)](#)
This method converts the supplied int to an OracleBoolean structure.
- [explicit operator OracleBoolean\(Int64\)](#)
This method converts the supplied Int64 to an OracleBoolean structure.
- [explicit operator OracleBoolean\(Single\)](#)
This method converts the supplied Single to an OracleBoolean structure.
- [explicit operator OracleBoolean\(String\)](#)
This method converts the supplied String to an OracleBoolean structure.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

explicit operator OracleBoolean(byte)

This method converts the supplied byte to an OracleBoolean structure.

Declaration

```
// C#  
public static explicit operator OracleBoolean(byte value1);
```

Parameters

- *value1*
A byte

Return Value

An OracleBoolean structure.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

explicit operator OracleBoolean(Decimal)

This method converts the supplied Decimal to an OracleBoolean structure.

Declaration

```
// C#  
public static explicit operator OracleBoolean(Decimal value1);
```

Parameters

- *value1*
A Decimal

Return Value

An OracleBoolean structure.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

explicit operator OracleBoolean(Double)

This method converts the supplied Double to an OracleBoolean structure.

Declaration

```
// C#  
public static explicit operator OracleBoolean(Double value1);
```

Parameters

- *value1*
A Double

Return Value

An OracleBoolean structure.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

explicit operator OracleBoolean(Int16)

This method converts the supplied Int16 to an OracleBoolean structure.

Declaration

```
// C#  
public static explicit operator OracleBoolean(Int16 value1);
```

Parameters

- *value1*
An Int16

Return Value

An OracleBoolean structure.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

explicit operator OracleBoolean(int)

This method converts the supplied int to an OracleBoolean structure.

Declaration

```
// C#  
public static explicit operator OracleBoolean(int value1);
```

Parameters

- *value1*
An int

Return Value

An OracleBoolean structure.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

explicit operator OracleBoolean(Int64)

This method converts the supplied Int64 to an OracleBoolean structure.

Declaration

```
// C#  
public static explicit operator OracleBoolean(Int64 value1);
```

Parameters

- *value1*
An Int64

Return Value

An OracleBoolean structure.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

explicit operator OracleBoolean(Single)

This method converts the supplied Single to an OracleBoolean structure.

Declaration

```
// C#  
public static explicit operator OracleBoolean(Single value1);
```

Parameters

- *value1*
A Single

Return Value

An OracleBoolean structure.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

explicit operator OracleBoolean(String)

This method converts the supplied String to an OracleBoolean structure.

Declaration

```
// C#  
public static explicit operator OracleBoolean(String value1);
```

Parameters

- *value1*
A String

Return Value

An OracleBoolean structure.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

OracleBoolean Properties

The OracleBoolean properties are listed in [Table 14-26](#).

Table 14-26 OracleBoolean Properties

Properties	Description
ByteValue	Returns a byte that represents the OracleBoolean structure
IsFalse	Indicates whether or not the value of the current instance is false
IsNull	Indicates whether or not the current instance has a null value
IsTrue	Indicates whether or not the value of the current instance is true
Value	Returns a boolean value that represents the current instance

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

ByteValue

This property returns a byte that represents the OracleBoolean structure.

Declaration

```
// C#  
public byte ByteValue {get;}
```

Property Value

A byte that represents the value of OracleBoolean structure.

Exceptions

OracleNullValueException – The current instance has a null value.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

IsFalse

This property indicates whether or not the value of the current instance is false.

Declaration

```
// C#  
public bool IsFalse {get;}
```

Property Value

A bool value that returns true if the current instance is false; otherwise, returns false.

Remarks

This property will return false if the current instance is null.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

IsNull

This property indicates whether or not the current instance has a null value.

Declaration

```
// C#  
public bool IsNull {get;}
```

Property Value

A bool value that returns true if the current instance has a null value; otherwise, returns false.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

IsTrue

This property indicates whether or not the value of the current instance is true.

Declaration

```
// C#  
public bool IsTrue {get;}
```

Property Value

A bool value that returns true if the current instance is true; otherwise, returns false.

Remarks

This property will return false if the current instance is null.

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

Value

This property returns a boolean value that represents the current instance.

Declaration

```
// C#  
public bool Value {get;}
```

Property Value

A bool value that returns true if the current instance is true; otherwise, returns false.

Exceptions

OracleNullValueException – The current instance has a null value.

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

OracleBoolean Instance Methods

The OracleBoolean instance methods are listed in [Table 14-27](#).

Table 14-27 OracleBoolean Instance Methods

Method	Description
CompareTo	Compares the current instance to the supplied object and returns an integer that represents their relative values
Equals	Determines whether or not an object is an instance of OracleBoolean, and whether or not the value of the object is equal to the current instance
GetHashCode	Returns a hash code for the current instance

Table 14-27 (Cont.) OracleBoolean Instance Methods

Method	Description
ToString	Returns the string representation of the current instance

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

CompareTo

This method compares the current instance to the supplied object and returns an integer that represents their relative values.

Declaration

```
// C#  
public int CompareTo(object obj);
```

Parameter

- *obj*
The supplied instance.

Return Value

The method returns a number:

- Less than zero: if the value of the current instance is less than *obj*.
- Zero: if the value of the current instance is equal to *obj*.
- Greater than zero: if the value of the current instance is greater than *obj*.

Implements

Comparable

Exceptions

ArgumentException - The parameter is not of type OracleBoolean.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between OracleBoolean. For example, comparing an OracleBoolean instance with an OracleBinary instance is not allowed. When an OracleBoolean is compared with a different type, an ArgumentException is thrown.

- Any OracleBoolean that has a value compares greater than an OracleBoolean that has a null value.
- Two OracleBoolean that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

Equals

Overrides Object

This method determines whether or not an object is an instance of OracleBoolean, and whether or not the value of the object is equal to the current instance.

Declaration

```
// C#  
public override bool Equals(object obj);
```

Parameter

- obj
An OracleBoolean instance.

Return Value

Returns true if obj is an instance of OracleBoolean, and the value of obj is equal to the current instance; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleBoolean that has a value compares greater than an OracleBoolean that has a null value.
- Two OracleBooleans that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

GetHashCode

Overrides Object

This method returns a hash code for the current instance.

Declaration

```
// C#  
public override int GetHashCode();
```

Return Value

Returns a hash code.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

ToString

Overrides Object

This method returns the string representation of the current instance.

Declaration

```
// C#  
public override string ToString();
```

Return Value

Returns the OracleBoolean value in a string representation.

Remarks

If the current instance has a null value, the returned string is null.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleBoolean Structure](#)
- [OracleBoolean Members](#)

OracleDate Structure

The OracleDate structure represents the Oracle DATE data type to be stored in or retrieved from a database. Each OracleDate stores the following information: year, month, day, hour, minute, and second.

Class Inheritance

System.Object
 System.ValueType
 Oracle.DataAccess.Types.OracleDate

Declaration

```
// C#  
public struct OracleDate : IComparable, INullable, IXmlSerializable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Types;  
using Oracle.DataAccess.Client;  
  
class OracleDateSample  
{  
    static void Main(string[] args)  
    {  
        // Initialize the dates to the lower and upper boundaries  
        OracleDate date1 = OracleDate.MinValue;  
        OracleDate date2 = OracleDate.MaxValue;  
        OracleDate date3 = new OracleDate(DateTime.MinValue);  
        OracleDate date4 = new OracleDate(DateTime.MaxValue);  
  
        // Set the thread's DateFormat for output  
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
```

```

info.DateFormat = "DD-MON-YYYY BC";
OracleGlobalization.SetThreadInfo(info);

// Print the lower and upper boundaries
Console.WriteLine("OracleDate ranges from\n{0}\nto\n{1}\n",
    date1, date2);
Console.WriteLine(".NET DateTime ranges from\n{0}\nto\n{1}\n",
    date3, date4);
}
}

```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Members](#)
- [OracleDate Constructors](#)
- [OracleDate Static Fields](#)
- [OracleDate Static Methods](#)
- [OracleDate Static Operators](#)
- [OracleDate Static Type Conversions](#)
- [OracleDate Properties](#)
- [OracleDate Methods](#)

OracleDate Members

OracleDate members are listed in the following tables:

OracleDate Constructors

OracleDate constructors are listed in [Table 14-28](#)

Table 14-28 OracleDate Constructors

Constructor	Description
OracleDate Constructors	Instantiates a new instance of OracleDate structure (Overloaded)

OracleDate Static Fields

The OracleDate static fields are listed in [Table 14-29](#).

Table 14-29 OracleDate Static Fields

Field	Description
MaxValue	Represents the maximum valid date for an OracleDate structure, which is December 31, 9999 23:59:59
MinValue	Represents the minimum valid date for an OracleDate structure, which is January 1, -4712 0:0:0

Table 14-29 (Cont.) OracleDate Static Fields

Field	Description
Null	Represents a null value that can be assigned to the value of an OracleDate structure instance

OracleDate Static Methods

The OracleDate static methods are listed in [Table 14-30](#).

Table 14-30 OracleDate Static Methods

Methods	Description
Equals	Determines if two OracleDate values are equal (Overloaded)
GreaterThan	Determines if the first of two OracleDate values is greater than the second
GreaterThanOrEqual	Determines if the first of two OracleDate values is greater than or equal to the second
LessThan	Determines if the first of two OracleDate values is less than the second
LessThanOrEqual	Determines if the first of two OracleDate values is less than or equal to the second
NotEquals	Determines if two OracleDate values are not equal
GetSysDate	Returns an OracleDate structure that represents the current date and time
Parse	Returns an OracleDate structure and sets its value using a string

OracleDate Static Operators

The OracleDate static operators are listed in [Table 14-31](#).

Table 14-31 OracleDate Static Operators

Operator	Description
operator ==	Determines if two OracleDate values are the same
operator >	Determines if the first of two OracleDate values is greater than the second
operator >=	Determines if the first of two OracleDate values is greater than or equal to the second
operator !=	Determines if the two OracleDate values are not equal
operator <	Determines if the first of two OracleDate values is less than the second
operator <=	Determines if the first of two OracleDate values is less than or equal to the second

OracleDate Static Type Conversions

The OracleDate static type conversions are listed in [Table 14-32](#).

Table 14-32 OracleDate Static Type Conversions

Operator	Description
explicit operator DateTime	Converts a structure to a DateTime structure
explicit operator OracleDate	Converts a structure to an OracleDate structure (Overloaded)

OracleDate Properties

The OracleDate properties are listed in [Table 14-33](#).

Table 14-33 OracleDate Properties

Properties	Description
BinData	Gets an array of bytes that represents an Oracle DATE in Oracle internal format
Day	Gets the day component of an OracleDate method
IsNull	Indicates whether or not the current instance has a null value
Hour	Gets the hour component of an OracleDate
Minute	Gets the minute component of an OracleDate
Month	Gets the month component of an OracleDate
Second	Gets the second component of an OracleDate
Value	Gets the date and time that is stored in the OracleDate structure
Year	Gets the year component of an OracleDate

OracleDate Methods

The OracleDate methods are listed in [Table 14-34](#).

Table 14-34 OracleDate Methods

Methods	Description
CompareTo	Compares the current OracleDate instance to an object, and returns an integer that represents their relative values
Equals	Determines whether or not an object has the same date and time as the current OracleDate instance (Overloaded)
GetHashCode	Returns a hash code for the OracleDate instance
GetDaysBetween	Calculates the number of days between the current OracleDate instance and an OracleDate structure
GetType	Inherited from System.Object
ToOracleTimeStamp	Converts the current OracleDate structure to an OracleTimeStamp structure
ToString	Converts the current OracleDate structure to a string

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)

OracleDate Constructors

The OracleDate constructors instantiates a new instance of the OracleDate structure.

Overload List:

- [OracleDate\(DateTime\)](#)
This constructor creates a new instance of the OracleDate structure and sets its value for date and time using the supplied DateTime value.
- [OracleDate\(string\)](#)
This constructor creates a new instance of the OracleDate structure and sets its value using the supplied string.
- [OracleDate\(int, int, int\)](#)
This constructor creates a new instance of the OracleDate structure and set its value for date using the supplied year, month, and day.
- [OracleDate\(int, int, int, int, int, int\)](#)
This constructor creates a new instance of the OracleDate structure and set its value for time using the supplied year, month, day, hour, minute, and second.
- [OracleDate\(byte \[\]\)](#)
This constructor creates a new instance of the OracleDate structure and sets its value to the provided byte array, which is in the internal Oracle DATE format.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

OracleDate(DateTime)

This constructor creates a new instance of the OracleDate structure and sets its value for date and time using the supplied DateTime value.

Declaration

```
// C#  
public OracleDate (DateTime dt);
```

Parameters

- *dt*
The provided DateTime value.

Remarks

The OracleDate structure only supports up to a second precision. The time value in the provided DateTime structure that has a precision smaller than second is ignored.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

OracleDate(string)

This constructor creates a new instance of the OracleDate structure and sets its value using the supplied string.

Declaration

```
// C#  
public OracleDate (string dateStr);
```

Parameters

- *dateStr*
A string that represents an Oracle DATE.

Exceptions

ArgumentException - The *dateStr* is an invalid string representation of an Oracle DATE or the *dateStr* is not in the date format specified by the thread's OracleGlobalization.DateFormat property, which represents the Oracle NLS_DATE_FORMAT parameter.

ArgumentNullException - The *dateStr* is null.

Remarks

The names and abbreviations used for months and days are in the language specified by the DateLanguage and Calendar properties of the thread's OracleGlobalization object. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Types;  
using Oracle.DataAccess.Client;
```

```
class OracleDateSample
{
    static void Main(string[] args)
    {
        // Set the thread's DateFormat for the OracleDate constructor
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.DateFormat = "YYYY-MON-DD";
        OracleGlobalization.SetThreadInfo(info);

        // construct OracleDate from a string using the DateFormat specified.
        OracleDate date = new OracleDate("1999-DEC-01");

        // Set a different DateFormat for the thread
        info.DateFormat = "MM/DD/YYYY";
        OracleGlobalization.SetThreadInfo(info);

        // Print "12/01/1999"
        Console.WriteLine(date.ToString());
    }
}
```

📘 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)
- ["OracleGlobalization Class"](#)
- *Oracle AI Database SQL Language Reference* for further information on date format elements

OracleDate(int, int, int)

This constructor creates a new instance of the `OracleDate` structure and set its value for date using the supplied year, month, and day.

Declaration

```
// C#
public OracleDate (int year, int month, int day);
```

Parameters

- *year*
The supplied year. Range of *year* is (-4712 to 9999).
- *month*
The supplied month. Range of *month* is (1 to 12).
- *day*
The supplied day. Range of *day* is (1 to 31).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleDate` (that is, the day is out of range for the month).

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

OracleDate(int, int, int, int, int, int)

This constructor creates a new instance of the `OracleDate` structure and set its value for time using the supplied year, month, day, hour, minute, and second.

Declaration

```
// C#  
public OracleDate (int year, int month, int day, int hour, int minute, int second);
```

Parameters

- *year*
The supplied year. Range of *year* is (-4712 to 9999).
- *month*
The supplied month. Range of *month* is (1 to 12).
- *day*
The supplied day. Range of *day* is (1 to 31).
- *hour*
The supplied hour. Range of *hour* is (0 to 23).
- *minute*
The supplied minute. Range of *minute* is (0 to 59).
- *second*
The supplied second. Range of *second* is (0 to 59).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleDate` (that is, the day is out of range for the month).

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

OracleDate(byte [])

This constructor creates a new instance of the `OracleDate` structure and sets its value to the provided byte array, which is in the internal Oracle DATE format.

Declaration

```
// C#  
public OracleDate(byte [] bytes);
```

Parameters

- *bytes*

A byte array that represents Oracle DATE in the internal Oracle DATE format.

Exceptions

`ArgumentException` - *bytes* is null or *bytes* is not in internal Oracle DATE format or *bytes* is not a valid Oracle DATE.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

OracleDate Static Fields

The `OracleDate` static fields are listed in [Table 14-35](#).

Table 14-35 OracleDate Static Fields

Field	Description
MaxValue	Represents the maximum valid date for an <code>OracleDate</code> structure, which is December 31, 9999 23:59:59
MinValue	Represents the minimum valid date for an <code>OracleDate</code> structure, which is January 1, -4712 0:0:0
Null	Represents a null value that can be assigned to the value of an <code>OracleDate</code> structure instance

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleDate Structure](#)
- [OracleDate Members](#)

MaxValue

This static field represents the maximum valid date for an `OracleDate` structure, which is December 31, 9999 23:59:59.

Declaration

```
// C#  
public static readonly OracleDate MaxValue;
```

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleDate Structure](#)
- [OracleDate Members](#)

MinValue

This static field represents the minimum valid date for an `OracleDate` structure, which is January 1, -4712.

Declaration

```
// C#  
public static readonly OracleDate MinValue;
```

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleDate Structure](#)
- [OracleDate Members](#)

Null

This static field represents a null value that can be assigned to the value of an `OracleDate` instance.

Declaration

```
// C#  
public static readonly OracleDate Null;
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

OracleDate Static Methods

The OracleDate static methods are listed in [Table 14-36](#).

Table 14-36 OracleDate Static Methods

Methods	Description
Equals	Determines if two OracleDate values are equal (Overloaded)
GreaterThan	Determines if the first of two OracleDate values is greater than the second
GreaterThanOrEqual	Determines if the first of two OracleDate values is greater than or equal to the second
LessThan	Determines if the first of two OracleDate values is less than the second
LessThanOrEqual	Determines if the first of two OracleDate values is less than or equal to the second
NotEquals	Determines if two OracleDate values are not equal
GetSysDate	Returns an OracleDate structure that represents the current date and time
Parse	Returns an OracleDate structure and sets its value using a string

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

Equals

Overloads Object

This method determines if two OracleDate values are equal.

Declaration

```
// C#  
public static bool Equals(OracleDate value1, OracleDate value2);
```

Parameters

- *value1*
The first OracleDate.
- *value2*
The second OracleDate.

Return Value

Returns true if two OracleDate values are equal; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleDate that has a value compares greater than an OracleDate that has a null value.
- Two OracleDates that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

GreaterThan

This method determines if the first of two OracleDate values is greater than the second.

Declaration

```
// C#  
public static bool GreaterThan(OracleDate value1, OracleDate value2);
```

Parameters

- *value1*
The first OracleDate.
- *value2*
The second OracleDate.

Return Value

Returns true if the first of two OracleDate values is greater than the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDate` that has a value compares greater than an `OracleDate` that has a null value.
- Two `OracleDates` that contain a null value are equal.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleDate Structure](#)
- [OracleDate Members](#)

GreaterThanOrEqual

This method determines if the first of two `OracleDate` values is greater than or equal to the second.

Declaration

```
// C#  
public static bool GreaterThanOrEqual(OracleDate value1, OracleDate value2);
```

Parameters

- *value1*
The first `OracleDate`.
- *value2*
The second `OracleDate`.

Return Value

Returns `true` if the first of two `OracleDate` values is greater than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDate` that has a value compares greater than an `OracleDate` that has a null value.
- Two `OracleDates` that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

LessThan

This method determines if the first of two `OracleDate` values is less than the second.

Declaration

```
// C#  
public static bool LessThan(OracleDate value1, OracleDate value2);
```

Parameters

- *value1*
The first `OracleDate`.
- *value2*
The second `OracleDate`.

Return Value

Returns `true` if the first of two `OracleDate` values is less than the second. Otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDate` that has a value compares greater than an `OracleDate` that has a null value.
- Two `OracleDates` that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

LessThanOrEqual

This method determines if the first of two `OracleDate` values is less than or equal to the second.

Declaration

```
// C#  
public static bool LessThanOrEqual(OracleDate value1, OracleDate value2);
```

Parameters

- *value1*
The first OracleDate.
- *value2*
The second OracleDate.

Return Value

Returns true if the first of two OracleDate values is less than or equal to the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleDate that has a value compares greater than an OracleDate that has a null value.
- Two OracleDates that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

NotEquals

This method determines if two OracleDate values are not equal.

Declaration

```
// C#  
public static bool NotEquals(OracleDate value1, OracleDate value2);
```

Parameters

- *value1*
The first OracleDate.
- *value2*
The second OracleDate.

Return Value

Returns true if two OracleDate values are not equal; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleDate that has a value compares greater than an OracleDate that has a null value.
- Two OracleDates that contain a null value are equal.

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleDate Structure](#)
- [OracleDate Members](#)

GetSysDate

This method gets an OracleDate structure that represents the current date and time.

Declaration

```
// C#  
public static OracleDate GetSysDate ();
```

Return Value

An OracleDate structure that represents the current date and time.

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleDate Structure](#)
- [OracleDate Members](#)

Parse

This method gets an OracleDate structure and sets its value for date and time using the supplied string.

Declaration

```
// C#  
public static OracleDate Parse (string dateStr);
```

Parameters

- *dateStr*
A string that represents an Oracle DATE.

Return Value

An OracleDate structure.

Exceptions

ArgumentException - The *dateStr* is an invalid string representation of an Oracle DATE or the *dateStr* is not in the date format specified by the thread's OracleGlobalization.DateFormat property, which represents the Oracle NLS_DATE_FORMAT parameter.

ArgumentNullException - The *dateStr* is null.

Remarks

The names and abbreviations used for months and days are in the language specified by the DateLanguage and Calendar properties of the thread's OracleGlobalization object. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#

using System;
using Oracle.DataAccess.Types;
using Oracle.DataAccess.Client;

class ParseSample
{
    static void Main(string[] args)
    {
        // Set the thread's DateFormat for the OracleDate constructor
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.DateFormat = "YYYY-MON-DD";
        OracleGlobalization.SetThreadInfo(info);

        // Construct OracleDate from a string using the DateFormat specified
        OracleDate date = OracleDate.Parse("1999-DEC-01");

        // Set a different DateFormat on the thread for ToString()
        info.DateFormat = "MM-DD-YY";
        OracleGlobalization.SetThreadInfo(info);

        // Print "12-01-1999"
        Console.WriteLine(date.ToString());
    }
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)
- *Oracle AI Database SQL Language Reference* for further information on datetime format elements

OracleDate Static Operators

The OracleDate static operators are listed in [Table 14-37](#).

Table 14-37 OracleDate Static Operators

Operator	Description
operator ==	Determines if two OracleDate values are the same
operator >	Determines if the first of two OracleDate values is greater than the second
operator >=	Determines if the first of two OracleDate values is greater than or equal to the second
operator !=	Determines if the two OracleDate values are not equal
operator <	Determines if the first of two OracleDate values is less than the second
operator <=	Determines if the first of two OracleDate values is less than or equal to the second

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

operator ==

This method determines if two OracleDate values are the same.

Declaration

```
// C#
public static bool operator == (OracleDate value1, OracleDate value2);
```

Parameters

- *value1*

The first `OracleDate`.

- `value2`

The second `OracleDate`.

Return Value

Returns `true` if they are the same; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDate` that has a value compares greater than an `OracleDate` that has a null value.
- Two `OracleDates` that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

operator >

This method determines if the first of two `OracleDate` values is greater than the second.

Declaration

```
// C#  
public static bool operator > (OracleDate value1, OracleDate value2);
```

Parameters

- `value1`

The first `OracleDate`.

- `value2`

The second `OracleDate`.

Return Value

Returns `true` if the first of two `OracleDate` values is greater than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDate` that has a value compares greater than an `OracleDate` that has a null value.
- Two `OracleDates` that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

operator >=

This method determines if the first of two `OracleDate` values is greater than or equal to the second.

Declaration

```
// C#  
public static bool operator >= (OracleDate value1, OracleDate value2);
```

Parameters

- *value1*
The first `OracleDate`.
- *value2*
The second `OracleDate`.

Return Value

Returns `true` if the first of two `OracleDate` values is greater than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDate` that has a value compares greater than an `OracleDate` that has a null value.
- Two `OracleDates` that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

operator !=

This method determines if the two `OracleDate` values are not equal.

Declaration

```
// C#  
public static bool operator != (OracleDate value1, OracleDate value2);
```

Parameters

- *value1*
The first OracleDate.
- *value2*
The second OracleDate.

Return Value

Returns true if the two OracleDate values are not equal; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleDate that has a value compares greater than an OracleDate that has a null value.
- Two OracleDates that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

operator <

This method determines if the first of two OracleDate values is less than the second.

Declaration

```
// C#  
public static bool operator < (OracleDate value1, OracleDate value2);
```

Parameters

- *value1*
The first OracleDate.
- *value2*
The second OracleDate.

Return Value

Returns true if the first of two OracleDate values is less than the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDate` that has a value compares greater than an `OracleDate` that has a null value.
- Two `OracleDates` that contain a null value are equal.

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleDate Structure](#)
- [OracleDate Members](#)

operator <=

This method determines if the first of two `OracleDate` values is less than or equal to the second.

Declaration

```
// C#  
public static bool operator <= (OracleDate value1, OracleDate value2);
```

Parameters

- *value1*
The first `OracleDate`.
- *value2*
The second `OracleDate`.

Return Value

Returns `true` if the first of two `OracleDate` values is less than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDate` that has a value compares greater than an `OracleDate` that has a null value.
- Two `OracleDates` that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

OracleDate Static Type Conversions

The OracleDate static type conversions are listed in [Table 14-38](#).

Table 14-38 OracleDate Static Type Conversions

Operator	Description
explicit operator DateTime	Converts a structure to a DateTime structure
explicit operator OracleDate	Converts a structure to an OracleDate structure (Overloaded)

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

explicit operator DateTime

This method converts an OracleDate structure to a DateTime structure.

Declaration

```
// C#
public static explicit operator DateTime(OracleDate val);
```

Parameters

- `val`
An OracleDate structure.

Return Value

A DateTime structure.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

explicit operator OracleDate

explicit operator OracleDate converts the provided structure to an OracleDate structure.

Overload List:

- [explicit operator OracleDate\(DateTime\)](#)
This method converts a DateTime structure to an OracleDate structure.
- [explicit operator OracleDate\(OracleTimeStamp\)](#)
This method converts an OracleTimeStamp structure to an OracleDate structure.
- [explicit operator OracleDate\(string\)](#)
This method converts the supplied string to an OracleDate structure.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

explicit operator OracleDate(DateTime)

This method converts a DateTime structure to an OracleDate structure.

Declaration

```
// C#  
public static explicit operator OracleDate(DateTime dt);
```

Parameters

- *dt*
A DateTime structure.

Return Value

An OracleDate structure.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

explicit operator OracleDate(OracleTimeStamp)

This method converts an `OracleTimeStamp` structure to an `OracleDate` structure.

Declaration

```
// C#  
public explicit operator OracleDate(OracleTimeStamp ts);
```

Parameters

- *ts*
`OracleTimeStamp`

Return Value

The returned `OracleDate` structure contains the date and time in the `OracleTimeStamp` structure.

Remarks

The precision of the `OracleTimeStamp` value can be lost during the conversion.

If the `OracleTimeStamp` structure has a null value, the returned `OracleDate` structure also has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

explicit operator OracleDate(string)

This method converts the supplied string to an `OracleDate` structure.

Declaration

```
// C#  
public explicit operator OracleDate (string dateStr);
```

Parameters

- *dateStr*

A string representation of an Oracle DATE.

Return Value

The returned `OracleDate` structure contains the date and time in the string *dateStr*.

Exceptions

`ArgumentNullException` - The *dateStr* is null.

`ArgumentException` - This exception is thrown if any of the following conditions exist:

- The *dateStr* is an invalid string representation of an Oracle DATE.
- The *dateStr* is not in the date format specified by the thread's `OracleGlobalization.DateFormat` property, which represents the Oracle NLS_DATE_FORMAT parameter.

Remarks

The names and abbreviations used for months and days are in the language specified by the `DateLanguage` and `Calendar` properties of the thread's `OracleGlobalization` object. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class OracleDateSample
{
    static void Main(string[] args)
    {
        // Set the thread's DateFormat to a specific format
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.DateFormat = "YYYY-MON-DD";
        OracleGlobalization.SetThreadInfo(info);

        // Construct OracleDate from a string using the DateFormat specified
        OracleDate date = (OracleDate)"1999-DEC-01";

        // Set a different DateFormat on the thread for ToString()
        info.DateFormat = "MON DD YY";
        OracleGlobalization.SetThreadInfo(info);

        // Prints "DEC 01 99"
        Console.WriteLine(date.ToString());
    }
}
```


① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)

OracleDate Properties

The OracleDate properties are listed in [Table 14-39](#).

Table 14-39 OracleDate Properties

Properties	Description
BinData	Gets an array of bytes that represents an Oracle DATE in Oracle internal format
Day	Gets the day component of an OracleDate method
IsNull	Indicates whether or not the current instance has a null value
Hour	Gets the hour component of an OracleDate
Minute	Gets the minute component of an OracleDate
Month	Gets the month component of an OracleDate
Second	Gets the second component of an OracleDate
Value	Gets the date and time that is stored in the OracleDate structure
Year	Gets the year component of an OracleDate

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

BinData

This property gets a array of bytes that represents an Oracle DATE in Oracle internal format.

Declaration

```
// C#  
public byte[] BinData{get;}
```

Property Value

An array of bytes.

Exceptions

OracleNullValueException - OracleDate has a null value.

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleDate Structure](#)
- [OracleDate Members](#)

Day

This property gets the day component of an OracleDate.

Declaration

```
// C#  
public int Day{get;}
```

Property Value

A number that represents the day. Range of Day is (1 to 31).

Exceptions

OracleNullValueException - OracleDate has a null value.

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleDate Structure](#)
- [OracleDate Members](#)

IsNull

This property indicates whether or not the current instance has a null value.

Declaration

```
// C#  
public bool IsNull{get;}
```

Property Value

Returns true if the current instance has a null value; otherwise, returns false.

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleDate Structure](#)
- [OracleDate Members](#)

Hour

This property gets the hour component of an `OracleDate`.

Declaration

```
// C#  
public int Hour {get;}
```

Property Value

A number that represents Hour. Range of Hour is (0 to 23).

Exceptions

`OracleNullValueException` - `OracleDate` has a null value.

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleDate Structure](#)
- [OracleDate Members](#)

Minute

This property gets the minute component of an `OracleDate`.

Declaration

```
// C#  
public int Minute {get;}
```

Property Value

A number that represents Minute. Range of Minute is (0 to 59).

Exceptions

`OracleNullValueException` - `OracleDate` has a null value.

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleDate Structure](#)
- [OracleDate Members](#)

Month

This property gets the month component of an `OracleDate`.

Declaration

```
// C#  
public int Month {get;}
```

Property Value

A number that represents Month. Range of Month is (1 to 12).

Exceptions

`OracleNullValueException` - `OracleDate` has a null value.

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleDate Structure](#)
- [OracleDate Members](#)

Second

This property gets the second component of an `OracleDate`.

Declaration

```
// C#  
public int Second {get;}
```

Property Value

A number that represents Second. Range of Second is (0 to 59).

Exceptions

`OracleNullValueException` - `OracleDate` has a null value.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleDate Structure](#)
- [OracleDate Members](#)

Value

This property specifies the date and time that is stored in the OracleDate structure.

Declaration

```
// C#  
public DateTime Value {get;}
```

Property Value

A DateTime.

Exceptions

OracleNullValueException - OracleDate has a null value.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleDate Structure](#)
- [OracleDate Members](#)

Year

This property gets the year component of an OracleDate.

Declaration

```
// C#  
public int Year {get;}
```

Property Value

A number that represents Year. Range of Year is (-4712 to 9999).

Exceptions

OracleNullValueException - OracleDate has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

OracleDate Methods

The OracleDate methods are listed in [Table 14-40](#).

Table 14-40 OracleDate Methods

Methods	Description
CompareTo	Compares the current OracleDate instance to an object, and returns an integer that represents their relative values
Equals	Determines whether or not an object has the same date and time as the current OracleDate instance (Overloaded)
GetHashCode	Returns a hash code for the OracleDate instance
GetDaysBetween	Calculates the number of days between the current OracleDate instance and an OracleDate structure
GetType	Inherited from System.Object
ToOracleTimeStamp	Converts the current OracleDate structure to an OracleTimeStamp structure
ToString	Converts the current OracleDate structure to a string

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

CompareTo

This method compares the current OracleDate instance to an object, and returns an integer that represents their relative values.

Declaration

```
// C#  
public int CompareTo(object obj);
```

Parameters

- *obj*
An object.

Return Value

The method returns:

- Less than zero: if the current `OracleDate` instance value is less than that of *obj*.
- Zero: if the current `OracleDate` instance and *obj* values are equal.
- Greater than zero: if the current `OracleDate` instance value is greater than *obj*.

Implements

`IComparable`

Exceptions

`ArgumentException` - The *obj* parameter is not an instance of `OracleDate`.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between `OracleDates`. For example, comparing an `OracleDate` instance with an `OracleBinary` instance is not allowed. When an `OracleDate` is compared with a different type, an `ArgumentException` is thrown.
- Any `OracleDate` that has a value compares greater than an `OracleDate` that has a null value.
- Two `OracleDates` that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

Equals

This method determines whether or not an object has the same date and time as the current `OracleDate` instance.

Declaration

```
// C#  
public override bool Equals( object obj);
```

Parameters

- *obj*
An object.

Return Value

Returns true if *obj* has the same type as the current instance and represents the same date and time; otherwise returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleDate that has a value compares greater than an OracleDate that has a null value.
- Two OracleDates that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

GetHashCode

Overrides Object

This method returns a hash code for the OracleDate instance.

Declaration

```
// C#  
public override int GetHashCode();
```

Return Value

A number that represents the hash code.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

GetDaysBetween

This method calculates the number of days between the current OracleDate instance and the supplied OracleDate structure.

Declaration

```
// C#  
public int GetDaysBetween (OracleDate val);
```


Parameters

- *val*
An OracleDate structure.

Return Value

The number of days between the current OracleDate instance and the OracleDate structure.

Exceptions

OracleNullValueException - The current instance or the supplied OracleDate structure has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

ToOracleTimeStamp

This method converts the current OracleDate structure to an OracleTimeStamp structure.

Declaration

```
// C#  
public OracleTimeStamp ToOracleTimeStamp();
```

Return Value

An OracleTimeStamp structure.

Remarks

The returned OracleTimeStamp structure has date and time in the current instance.

If the OracleDate instance has a null value, the returned OracleTimeStamp structure has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)

ToString

Overrides ValueType

This method converts the current OracleDate structure to a string.

Declaration

```
// C#  
public override string ToString();
```

Return Value

A string.

Remarks

The returned value is a string representation of the OracleDate in the format specified by the thread's OracleGlobalization.DateFormat property. The names and abbreviations used for months and days are in the language specified by the thread's OracleGlobalization.DateLanguage and OracleGlobalization.Calendar properties. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Client;  
using Oracle.DataAccess.Types;  
  
class ToStringSample  
{  
    static void Main(string[] args)  
    {  
        // Set the thread's DateFormat to a specific format  
        OracleGlobalization info = OracleGlobalization.GetClientInfo();  
        info.DateFormat = "YYYY-MON-DD";  
        OracleGlobalization.SetThreadInfo(info);  
  
        // Construct OracleDate from a string using the DateFormat specified  
        OracleDate date = (OracleDate)"1999-DEC-01";  
  
        // Set a different DateFormat on the thread for ToString()  
        info.DateFormat = "YYYY/MM/DD";  
        OracleGlobalization.SetThreadInfo(info);  
  
        // Prints "1999/12/01"  
        Console.WriteLine(date.ToString());  
    }  
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDate Structure](#)
- [OracleDate Members](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)

OracleDecimal Structure

The OracleDecimal structure represents an Oracle NUMBER in the database or any Oracle numeric value.

Class Inheritance

```
System.Object
    System.ValueType
        Oracle.DataAccess.Types.OracleDecimal
```

Declaration

```
// C#
public struct OracleDecimal : IComparable, INullable, IXmlSerializable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

OracleDecimal can store up to 38 precision, while the .NET Decimal data type can only hold up to 28 precision. When accessing the OracleDecimal.Value property from an OracleDecimal that has a value greater than 28 precision, an exception is thrown. To retrieve the actual value of OracleDecimal, use the OracleDecimal.ToString() method. Another

approach is to obtain the OracleDecimal value as a byte array in an internal Oracle NUMBER format through the BinData property.

Example

```
// C#

using System;
using Oracle.DataAccess.Types;

class OracleDecimalSample
{
    static void Main(string[] args)
    {
        // Illustrates the range of OracleDecimal vs. .NET decimal
        OracleDecimal decimal1 = OracleDecimal.MinValue;
        OracleDecimal decimal2 = OracleDecimal.MaxValue;
        OracleDecimal decimal3 = new OracleDecimal(decimal.MinValue);
        OracleDecimal decimal4 = new OracleDecimal(decimal.MaxValue);

        // Print the ranges
        Console.WriteLine("OracleDecimal can range from\n{0}\nto\n{1}\n",
            decimal1, decimal2);
        Console.WriteLine(".NET decimal can range from\n{0}\nto\n{1}\n",
            decimal3, decimal4);
    }
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Constructors](#)
- [OracleDecimal Static Fields](#)
- [OracleDecimal Static \(Comparison\) Methods](#)
- [OracleDecimal Static \(Manipulation\) Methods](#)
- [OracleDecimal Static \(Logarithmic\) Methods](#)
- [OracleDecimal Static \(Trigonometric\) Methods](#)
- [OracleDecimal Static \(Comparison\) Operators](#)
- [OracleDecimal Static Operators \(Conversion from .NET Type to OracleDecimal\)](#)
- [OracleDecimal Static Operators \(Conversion from OracleDecimal to .NET\)](#)
- [OracleDecimal Properties](#)
- [OracleDecimal Instance Methods](#)

OracleDecimal Members

OracleDecimal members are listed in the following tables:

OracleDecimal Constructors

OracleDecimal constructors are listed in [Table 14-41](#)

Table 14-41 OracleDecimal Constructors

Constructor	Description
OracleDecimal Constructors	Instantiates a new instance of OracleDecimal structure (Overloaded)

OracleDecimal Static Fields

The OracleDecimal static fields are listed in [Table 14-42](#).

Table 14-42 OracleDecimal Static Fields

Field	Description
MaxPrecision	A constant representing the maximum precision, which is 38
MaxScale	A constant representing the maximum scale, which is 127
MaxValue	A constant representing the maximum value for this structure, which is $9.9...9 \times 10^{125}$
MinScale	A constant representing the minimum scale, which is -84
MinValue	A constant representing the minimum value for this structure, which is -1.0×10^{130}
NegativeOne	A constant representing the negative one value
Null	Represents a null value that can be assigned to an OracleDecimal instance
One	A constant representing the positive one value
Pi	A constant representing the numeric Pi value
Zero	A constant representing the zero value

OracleDecimal Static (Comparison) Methods

The OracleDecimal static (comparison) methods are listed in [Table 14-43](#).

Table 14-43 OracleDecimal Static (Comparison) Methods

Methods	Description
Equals	Determines if two OracleDecimal values are equal (Overloaded)
GreaterThan	Determines if the first of two OracleDecimal values is greater than the second
GreaterThanOrEqual	Determines if the first of two OracleDecimal values is greater than or equal to the second
LessThan	Determines if the first of two OracleDecimal values is less than the second
LessThanOrEqual	Determines if the first of two OracleDecimal values is less than or equal to the second.
NotEquals	Determines if two OracleDecimal values are not equal

OracleDecimal Static (Manipulation) Methods

The OracleDecimal static (manipulation) methods are listed in [Table 14-44](#).

Table 14-44 OracleDecimal Static (Manipulation) Methods

Methods	Description
Abs	Returns the absolute value of an OracleDecimal
Add	Adds two OracleDecimal structures
AdjustScale	Returns a new OracleDecimal with the specified number of digits and indicates whether or not to round or truncate the number if the scale is less than original
Ceiling	Returns a new OracleDecimal structure with its value set to the ceiling of an OracleDecimal structure
ConvertToPrecScale	Returns a new OracleDecimal structure with a new precision and scale
Divide	Divides one OracleDecimal value by another
Floor	Returns a new OracleDecimal structure with its value set to the floor of an OracleDecimal structure
Max	Returns the maximum value of the two supplied OracleDecimal structures
Min	Returns the minimum value of the two supplied OracleDecimal structures
Mod	Returns a new OracleDecimal structure with its value set to the modulus of two OracleDecimal structures
Multiply	Returns a new OracleDecimal structure with its value set to the result of multiplying two OracleDecimal structures
Negate	Returns a new OracleDecimal structure with its value set to the negation of the supplied OracleDecimal structure
Parse	Converts a string to an OracleDecimal
Round	Returns a new OracleDecimal structure with its value set to that of the supplied OracleDecimal structure and rounded off to the specified place
SetPrecision	Returns a new OracleDecimal structure with a new specified precision.
Shift	Returns a new OracleDecimal structure with its value set to that of the supplied OracleDecimal structure, and its decimal place shifted to the specified number of places to the right
Sign	Determines the sign of an OracleDecimal structure
Sqrt	Returns a new OracleDecimal structure with its value set to the square root of the supplied OracleDecimal structure
Subtract	Returns a new OracleDecimal structure with its value set to result of subtracting one OracleDecimal structure from another
Truncate	Truncates the OracleDecimal at a specified position

OracleDecimal Static (Logarithmic) Methods

The OracleDecimal static (logarithmic) methods are listed in [Table 14-45](#).

Table 14-45 OracleDecimal Static (Logarithmic) Methods

Methods	Description
Exp	Returns a new OracleDecimal structure with its value set to e raised to the supplied power
Log	Returns the supplied OracleDecimal structure with its value set to the logarithm of the supplied OracleDecimal structure (Overloaded)
Pow	Returns a new OracleDecimal structure with its value set to the supplied OracleDecimal structure raised to the supplied power (Overloaded)

OracleDecimal Static (Trigonometric) Methods

The OracleDecimal static (trigonometric) methods are listed in [Table 14-46](#).

Table 14-46 OracleDecimal Static (Trigonometric) Methods

Methods	Description
Acos	Returns an angle in radians whose cosine is the supplied OracleDecimal structure
Asin	Returns an angle in radians whose sine is the supplied OracleDecimal structure
Atan	Returns an angle in radians whose tangent is the supplied OracleDecimal structure
Atan2	Returns an angle in radians whose tangent is the quotient of the two supplied OracleDecimal structures
Cos	Returns the cosine of the supplied angle in radians
Sin	Returns the sine of the supplied angle in radians
Tan	Returns the tangent of the supplied angle in radians
Cosh	Returns the hyperbolic cosine of the supplied angle in radians
Sinh	Returns the hyperbolic sine of the supplied angle in radians
Tanh	Returns the hyperbolic tangent of the supplied angle in radians

OracleDecimal Static (Comparison) Operators

The OracleDecimal static (comparison) operators are listed in [Table 14-47](#).

Table 14-47 OracleDecimal Static (Comparison) Operators

Operator	Description
operator +	Adds two OracleDecimal values
operator /	Divides one OracleDecimal value by another
operator ==	Determines if the two OracleDecimal values are equal
operator >	Determines if the first of two OracleDecimal values is greater than the second
operator >=	Determines if the first of two OracleDecimal values is greater than or equal to the second

Table 14-47 (Cont.) OracleDecimal Static (Comparison) Operators

Operator	Description
operator !=	Determines if the two OracleDecimal values are not equal
operator <	Determines if the first of two OracleDecimal values is less than the second
operator <=	Determines if the first of two OracleDecimal values is less than or equal to the second
operator *	Multiplies two OracleDecimal structures
operator -	Subtracts one OracleDecimal structure from another
operator -	Negates an OracleDecimal structure
operator %	Returns a new OracleDecimal structure with its value set to the modulus of two OracleDecimal structures.

OracleDecimal Static Operators (Conversion from .NET Type to OracleDecimal)

The OracleDecimal static operators (Conversion from .NET Type to OracleDecimal) are listed in [Table 14-48](#).

Table 14-48 OracleDecimal Static Operators (Conversion from .NET Type to OracleDecimal)

Operator	Description
implicit operator OracleDecimal	Converts an instance value to an OracleDecimal structure (Overloaded)
explicit operator OracleDecimal	Converts an instance value to an OracleDecimal structure (Overloaded)

OracleDecimal Static Operators (Conversion from OracleDecimal to .NET)

The OracleDecimal static operators (Conversion from OracleDecimal to .NET) are listed in [Table 14-49](#).

Table 14-49 OracleDecimal Static Operators (Conversion from OracleDecimal to .NET)

Operator	Description
explicit operator byte	Returns the byte representation of the OracleDecimal value
explicit operator decimal	Returns the decimal representation of the OracleDecimal value
explicit operator double	Returns the double representation of the OracleDecimal value
explicit operator short	Returns the short representation of the OracleDecimal value
explicit operator int	Returns the int representation of the OracleDecimal value
explicit operator long	Returns the long representation of the OracleDecimal value
explicit operator float	Returns the float representation of the OracleDecimal value

OracleDecimal Properties

The OracleDecimal properties are listed in [Table 14-50](#).

Table 14-50 OracleDecimal Properties

Properties	Description
BinData	Returns a byte array that represents the Oracle NUMBER in Oracle internal format
Format	Specifies the format for ToString()
IsInt	Indicates whether or not the current instance is an integer
IsNull	Indicates whether or not the current instance has a null value
IsPositive	Indicates whether or not the current instance is greater than 0
IsZero	Indicates whether or not the current instance has a zero value
Value	Returns a decimal value

OracleDecimal Instance Methods

The OracleDecimal instance methods are listed in [Table 14-51](#).

Table 14-51 OracleDecimal Instance Methods

Method	Description
CompareTo	Compares the current instance to the supplied object and returns an integer that represents their relative values
Equals	Determines whether or not an object is an instance of OracleDecimal, and whether or not the value of the object is equal to the current instance (Overloaded)
GetHashCode	Returns a hash code for the current instance
GetType	Inherited from System.Object
ToByte	Returns the byte representation of the current instance
ToDouble	Returns the double representation of the current instance
ToInt16	Returns the Int16 representation of the current instance
ToInt32	Returns the Int32 representation of the current instance
ToInt64	Returns the Int64 representation of the current instance
ToSingle	Returns the Single representation of the current instance
ToString	Overloads Object.ToString() Returns the string representation of the current instance

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Structure](#)

OracleDecimal Constructors

The OracleDecimal constructors instantiate a new instance of the OracleDecimal structure.

Overload List:

- [OracleDecimal\(byte \[\]\)](#)
This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied byte array, which is in an Oracle NUMBER format.
- [OracleDecimal\(decimal\)](#)
This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied `Decimal` value.
- [OracleDecimal\(double\)](#)
This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied double value.
- [OracleDecimal\(int\)](#)
This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied `Int32` value.
- [OracleDecimal\(float\)](#)
This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied `Single` value.
- [OracleDecimal\(long\)](#)
This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied `Int64` value.
- [OracleDecimal\(string\)](#)
This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied string value.
- [OracleDecimal\(string, string\)](#)
This constructor creates a new instance of the `OracleDecimal` structure with the supplied string value and number format.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

OracleDecimal(byte [])

This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied byte array, which is in an Oracle NUMBER format.

Declaration

```
// C#  
public OracleDecimal(byte [] bytes);
```

Parameters

- *bytes*

A byte array that represents an Oracle NUMBER in an internal Oracle format.

Exceptions

ArgumentException - The *bytes* parameter is not in a internal Oracle NUMBER format or *bytes* has an invalid value.

ArgumentNullException - The *bytes* parameter is null.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

OracleDecimal(decimal)

This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied `Decimal` value.

Declaration

```
// C#  
public OracleDecimal(decimal decX);
```

Parameters

- *decX*

The provided `Decimal` value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

OracleDecimal(double)

This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied `double` value.

Declaration

```
// C#  
public OracleDecimal(double doubleX)
```

Parameters

- *doubleX*
The provided double value.

Exceptions

OverflowException - The value of the supplied double is greater than the maximum value or less than the minimum value of OracleDecimal.

Remarks

OracleDecimal contains the following values depending on the provided double value:

- double.PositiveInfinity: positive infinity value
- double.NegativeInfinity: negative infinity value.
- double.NaN: null value

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

OracleDecimal(int)

This constructor creates a new instance of the OracleDecimal structure and sets its value to the supplied Int32 value.

Declaration

```
// C#  
public OracleDecimal(int intX);
```

Parameters

- *intX*
The provided Int32 value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

OracleDecimal(float)

This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied `Single` value.

Declaration

```
// C#  
public OracleDecimal(float floatX);
```

Parameters

- *floatX*
The provided float value.

Remarks

`OracleDecimal` contains the following values depending on the provided float value:

`float.PositiveInfinity`: positive infinity value

`float.NegativeInfinity`: negative infinity value

`float.NaN`: null value

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

OracleDecimal(long)

This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied `Int64` value.

Declaration

```
// C#  
public OracleDecimal(long longX);
```

Parameters

- *longX*
The provided `Int64` value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

OracleDecimal(string)

This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied string value.

Declaration

```
// C#  
public OracleDecimal(string numStr);
```

Parameters

- *numStr*
The provided string value.

Exceptions

`ArgumentException` - The *numStr* parameter is an invalid string representation of an `OracleDecimal`.

`ArgumentNullException` - The *numStr* parameter is null.

`OverflowException` - The value of *numStr* is greater than the maximum value or less than the minimum value of `OracleDecimal`.

input string format is incorrect - The locale's numeric separator is a comma(,).

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)

OracleDecimal(string, string)

This constructor creates a new instance of the `OracleDecimal` structure with the supplied string value and number format.

Declaration

```
// C#  
public OracleDecimal(string numStr, string format);
```

Parameters

- *numStr*
The provided string value.
- *format*
The provided number format.

Exceptions

ArgumentException - The *numStr* parameter is an invalid string representation of an *OracleDecimal* or the *numStr* is not in the numeric format specified by *format*.

ArgumentNullException - The *numStr* parameter is null.

OverflowException - The value of *numStr* parameter is greater than the maximum value or less than the minimum value of *OracleDecimal*.

Remarks

If the numeric format includes decimal and group separators, then the provided string must use those characters defined by the *OracleGlobalization.NumericCharacters* of the thread.

If the numeric format includes the currency symbol, ISO currency symbol, or the dual currency symbol, then the provided string must use those symbols defined by the *OracleGlobalization.Currency*, *OracleGlobalization.ISOCurrency*, and *OracleGlobalization.DualCurrency* properties respectively.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Client;  
using Oracle.DataAccess.Types;  
  
class OracleDecimalSample  
{  
    static void Main(string[] args)  
    {  
        // Set the nls parameters related to currency  
        OracleGlobalization info = OracleGlobalization.GetClientInfo();  
        info.Currency = "$";  
        info.NumericCharacters = ".,";  
        OracleGlobalization.SetThreadInfo(info);  
  
        // Construct an OracleDecimal using a valid numeric format  
        OracleDecimal dec = new OracleDecimal("$2,222.22", "L9G999D99");  
  
        // Print "$2,222.22"  
        Console.WriteLine(dec.ToString());  
    }  
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)

OracleDecimal Static Fields

The OracleDecimal static fields are listed in [Table 14-52](#).

Table 14-52 OracleDecimal Static Fields

Field	Description
MaxPrecision	A constant representing the maximum precision, which is 38
MaxScale	A constant representing the maximum scale, which is 127
MaxValue	A constant representing the maximum value for this structure, which is 9.9...9 x 10 ¹²⁵
MinScale	A constant representing the minimum scale, which is -84
MinValue	A constant representing the minimum value for this structure, which is -1.0 x 10 ¹³⁰
NegativeOne	A constant representing the negative one value
Null	Represents a null value that can be assigned to an OracleDecimal instance
One	A constant representing the positive one value
Pi	A constant representing the numeric Pi value
Zero	A constant representing the zero value

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

MaxPrecision

This static field represents the maximum precision, which is 38.

Declaration

```
// C#
public static readonly byte MaxPrecision;
```


① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

MaxScale

This static field a constant representing the maximum scale, which is 127.

Declaration

```
// C#  
public static readonly byte MaxScale;
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

MaxValue

This static field indicates a constant representing the maximum value for this structure, which is $9.9...9 \times 10^{125}$ (38 nines followed by 88 zeroes).

Declaration

```
// C#  
public static readonly OracleDecimal MaxValue;
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

MinScale

This static field indicates a constant representing the maximum scale, which is -84.

Declaration

```
// C#  
public static readonly int MinScale;
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

MinValue

This static field indicates a constant representing the minimum value for this structure, which is -1.0×10^{130} .

Declaration

```
// C#  
public static readonly OracleDecimal MinValue;
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

NegativeOne

This static field indicates a constant representing the negative one value.

Declaration

```
// C#  
public static readonly OracleDecimal NegativeOne;
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Null

This static field represents a null value that can be assigned to an OracleDecimal instance.

Declaration

```
// C#  
public static readonly OracleDecimal Null;
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

One

This static field indicates a constant representing the positive one value.

Declaration

```
// C#  
public static readonly OracleDecimal One;
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Pi

This static field indicates a constant representing the numeric Pi value.

Declaration

```
// C#  
public static readonly OracleDecimal Pi;
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Zero

This static field indicates a constant representing the zero value.

Declaration

```
// C#  
public static readonly OracleDecimal Zero;
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

OracleDecimal Static (Comparison) Methods

The OracleDecimal static (comparison) methods are listed in [Table 14-53](#).

Table 14-53 OracleDecimal Static (Comparison) Methods

Methods	Description
Equals	Determines if two OracleDecimal values are equal (Overloaded)
GreaterThan	Determines if the first of two OracleDecimal values is greater than the second
GreaterThanOrEqual	Determines if the first of two OracleDecimal values is greater than or equal to the second
LessThan	Determines if the first of two OracleDecimal values is less than the second
LessThanOrEqual	Determines if the first of two OracleDecimal values is less than or equal to the second.
NotEquals	Determines if two OracleDecimal values are not equal

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Equals

This method determines if two OracleDecimal values are equal.

Declaration

```
// C#
public static bool Equals(OracleDecimal value1, OracleDecimal value2);
```

Parameters

- *value1*
The first OracleDecimal.
- *value2*

The second `OracleDecimal`.

Return Value

Returns true if two `OracleDecimal` values are equal; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDecimal` that has a value compares greater than an `OracleDecimal` that has a null value.
- Two `OracleDecimals` that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

GreaterThan

This method determines if the first of two `OracleDecimal` values is greater than the second.

Declaration

```
// C#  
public static bool GreaterThan(OracleDecimal value1, OracleDecimal value2);
```

Parameters

- *value1*
The first `OracleDecimal`.
- *value2*
The second `OracleDecimal`.

Return Value

Returns true if the first of two `OracleDecimal` values is greater than the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDecimal` that has a value compares greater than an `OracleDecimal` that has a null value.
- Two `OracleDecimals` that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

GreaterThanOrEqual

This method determines if the first of two `OracleDecimal` values is greater than or equal to the second.

Declaration

```
// C#  
public static bool GreaterThanOrEqual(OracleDecimal value1, OracleDecimal value2);
```

Parameters

- *value1*
The first `OracleDecimal`.
- *value2*
The second `OracleDecimal`.

Return Value

Returns `true` if the first of two `OracleDecimal` values is greater than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDecimal` that has a value compares greater than an `OracleDecimal` that has a null value.
- Two `OracleDecimals` that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

LessThan

This method determines if the first of two `OracleDecimal` values is less than the second.

Declaration

```
// C#  
public static bool LessThan(OracleDecimal value1, OracleDecimal value2);
```

Parameters

- *value1*
The first OracleDecimal.
- *value2*
The second OracleDecimal.

Return Value

Returns true if the first of two OracleDecimal values is less than the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleDecimal that has a value compares greater than an OracleDecimal that has a null value.
- Two OracleDecimals that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

LessThanOrEqual

This method determines if the first of two OracleDecimal values is less than or equal to the second.

Declaration

```
// C#  
public static bool LessThanOrEqual(OracleDecimal value1, OracleDecimal value2);
```

Parameters

- *value1*
The first OracleDecimal.
- *value2*
The second OracleDecimal.

Return Value

Returns true if the first of two `OracleDecimal` values is less than or equal to the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDecimal` that has a value compares greater than an `OracleDecimal` that has a null value.
- Two `OracleDecimals` that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

NotEquals

This method determines if two `OracleDecimal` values are not equal.

Declaration

```
// C#  
public static bool NotEquals(OracleDecimal value1, OracleDecimal value2);
```

Parameters

- *value1*
The first `OracleDecimal`.
- *value2*
The second `OracleDecimal`.

Return Value

Returns true if two `OracleDecimal` values are not equal; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDecimal` that has a value compares greater than an `OracleDecimal` that has a null value.
- Two `OracleDecimals` that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

OracleDecimal Static (Manipulation) Methods

The OracleDecimal static (manipulation) methods are listed in [Table 14-54](#).

Table 14-54 OracleDecimal Static (Manipulation) Methods

Methods	Description
Abs	Returns the absolute value of an OracleDecimal
Add	Adds two OracleDecimal structures
AdjustScale	Returns a new OracleDecimal with the specified number of digits and indicates whether or not to round or truncate the number if the scale is less than original
Ceiling	Returns a new OracleDecimal structure with its value set to the ceiling of an OracleDecimal structure
ConvertToPrecScale	Returns a new OracleDecimal structure with a new precision and scale
Divide	Divides one OracleDecimal value by another
Floor	Returns a new OracleDecimal structure with its value set to the floor of an OracleDecimal structure
Max	Returns the maximum value of the two supplied OracleDecimal structures
Min	Returns the minimum value of the two supplied OracleDecimal structures
Mod	Returns a new OracleDecimal structure with its value set to the modulus of two OracleDecimal structures
Multiply	Returns a new OracleDecimal structure with its value set to the result of multiplying two OracleDecimal structures
Negate	Returns a new OracleDecimal structure with its value set to the negation of the supplied OracleDecimal structure
Parse	Converts a string to an OracleDecimal
Round	Returns a new OracleDecimal structure with its value set to that of the supplied OracleDecimal structure and rounded off to the specified place
SetPrecision	Returns a new OracleDecimal structure with a new specified precision.
Shift	Returns a new OracleDecimal structure with its value set to that of the supplied OracleDecimal structure, and its decimal place shifted to the specified number of places to the right
Sign	Determines the sign of an OracleDecimal structure

Table 14-54 (Cont.) OracleDecimal Static (Manipulation) Methods

Methods	Description
Sqrt	Returns a new OracleDecimal structure with its value set to the square root of the supplied OracleDecimal structure
Subtract	Returns a new OracleDecimal structure with its value set to result of subtracting one OracleDecimal structure from another
Truncate	Truncates the OracleDecimal at a specified position

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Abs

This method returns the absolute value of an OracleDecimal.

Declaration

```
// C#  
public static OracleDecimal Abs(OracleDecimal val);
```

Parameters

- *val*
An OracleDecimal.

Return Value

The absolute value of an OracleDecimal.

Remarks

If either argument has a null value, the returned OracleDecimal has a null value.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Add

This method adds two OracleDecimal structures.

Declaration

```
// C#  
public static OracleDecimal Add(OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*
The first OracleDecimal.
- *val2*
The second OracleDecimal.

Return Value

Returns an OracleDecimal structure.

Remarks

If either argument has a null value, the returned OracleDecimal has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

AdjustScale

This method returns a new OracleDecimal with the specified number of digits and indicates whether or not to round or truncate the number if the scale is less than the original.

Declaration

```
// C#  
public static OracleDecimal AdjustScale(OracleDecimal val, int digits,  
    bool fRound);
```

Parameters

- *val*
An OracleDecimal.
- *digits*
The number of digits.
- *fRound*
Indicates whether or not to round or truncate the number. Setting it to true rounds the number and setting it to false truncates the number.

Return Value

An OracleDecimal.

Remarks

If the supplied `OracleDecimal` has a null value, the returned `OracleDecimal` has a null value.

Example

```
// C#

using System;
using Oracle.DataAccess.Types;

class AdjustScaleSample
{
    static void Main(string[] args)
    {
        OracleDecimal dec1 = new OracleDecimal(5.555);

        // Adjust Scale to 2 with rounding off
        OracleDecimal dec2 = OracleDecimal.AdjustScale(dec1, 2, true);

        // Prints 5.56
        Console.WriteLine(dec2.ToString());

        // Adjust Scale to 2 with truncation
        OracleDecimal dec3 = OracleDecimal.AdjustScale(dec1, 2, false);

        // Prints 5.55
        Console.WriteLine(dec3.ToString());
    }
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Ceiling

This method returns a new `OracleDecimal` structure with its value set to the ceiling of the supplied `OracleDecimal`.

Declaration

```
// C#
public static OracleDecimal Ceiling(OracleDecimal val);
```

Parameters

- `val`
An `OracleDecimal`.

Return Value

A new `OracleDecimal` structure.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

ConvertToPrecScale

This method returns a new `OracleDecimal` structure with a new precision and scale.

Declaration

```
// C#
public static OracleDecimal ConvertToPrecScale(OracleDecimal val
    int precision, int scale);
```

Parameters

- *val*
An `OracleDecimal` structure.
- *precision*
The precision. Range of precision is 1 to 38.
- *scale*
The number of digits to the right of the decimal point. Range of scale is -84 to 127.

Return Value

A new `OracleDecimal` structure.

Remarks

If the supplied `OracleDecimal` has a null value, the returned `OracleDecimal` has a null value.

Example

```
// C#

using System;
using Oracle.DataAccess.Types;

class ConvertToPrecScaleSample
{
    static void Main(string[] args)
    {
        OracleDecimal dec1 = new OracleDecimal(555.6666);

        // Set the precision of od to 5 and scale to 2
    }
}
```

```
OracleDecimal dec2 = OracleDecimal.ConvertToPrecScale(dec1,5,2);

// Prints 555.67
Console.WriteLine(dec2.ToString());

// Set the precision of od to 3 and scale to 0
OracleDecimal dec3 = OracleDecimal.ConvertToPrecScale(dec1,3,0);

// Prints 556
Console.WriteLine(dec3.ToString());
    }
}
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Divide

This method divides one `OracleDecimal` value by another.

Declaration

```
// C#
public static OracleDecimal Divide(OracleDecimal val1, OracleDecimal val2);
```

Parameters

- `val1`
An `OracleDecimal`.
- `val2`
An `OracleDecimal`.

Return Value

A new `OracleDecimal` structure.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Floor

This method returns a new `OracleDecimal` structure with its value set to the floor of the supplied `OracleDecimal` structure.

Declaration

```
// C#  
public static OracleDecimal Floor(OracleDecimal val);
```

Parameters

- *val*
An `OracleDecimal` structure.

Return Value

A new `OracleDecimal` structure.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Max

This method returns the maximum value of the two supplied `OracleDecimal` structures.

Declaration

```
// C#  
public static OracleDecimal Max(OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*
An `OracleDecimal` structure.
- *val2*
An `OracleDecimal` structure.

Return Value

An `OracleDecimal` structure that has the greater value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Min

This method returns the minimum value of the two supplied OracleDecimal structures.

Declaration

```
// C#  
public static OracleDecimal Min(OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*
An OracleDecimal structure.
- *val2*
An OracleDecimal structure.

Return Value

An OracleDecimal structure that has the smaller value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Mod

This method returns a new OracleDecimal structure with its value set to the modulus of two OracleDecimal structures.

Declaration

```
// C#  
public static OracleDecimal Mod(OracleDecimal val1, OracleDecimal divider);
```

Parameters

- *val1*
An OracleDecimal structure.
- *divider*

An OracleDecimal structure.

Return Value

An OracleDecimal.

Remarks

If either argument has a null value, the returned OracleDecimal has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Multiply

This method returns a new OracleDecimal structure with its value set to the result of multiplying two OracleDecimal structures.

Declaration

```
// C#  
public static OracleDecimal Multiply(OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*
An OracleDecimal structure.
- *val2*
An OracleDecimal structure.

Return Value

A new OracleDecimal structure.

Remarks

If either argument has a null value, the returned OracleDecimal has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Negate

This method returns a new `OracleDecimal` structure with its value set to the negation of the supplied `OracleDecimal` structures.

Declaration

```
// C#  
public static OracleDecimal Negate(OracleDecimal val);
```

Parameters

- *val*
An `OracleDecimal` structure.

Return Value

A new `OracleDecimal` structure.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Parse

This method converts a string to an `OracleDecimal`.

Declaration

```
// C#  
public static OracleDecimal Parse (string str);
```

Parameters

- *str*
The string being converted.

Return Value

A new `OracleDecimal` structure.

Exceptions

`ArgumentException` - The *numStr* parameter is an invalid string representation of an `OracleDecimal`.

`ArgumentNullException` - The *numStr* parameter is null.

OverflowException - The value of *numStr* is greater than the maximum value or less than the minimum value of OracleDecimal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)

Round

This method returns a new OracleDecimal structure with its value set to that of the supplied OracleDecimal structure and rounded off to the specified place.

Declaration

```
// C#  
public static OracleDecimal Round(OracleDecimal val, int decplace);
```

Parameters

- *val*
An OracleDecimal structure.
- *decplace*
The specified decimal place. If the value is positive, the function rounds the OracleDecimal structure to the right of the decimal point. If the value is negative, the function rounds to the left of the decimal point.

Return Value

An OracleDecimal structure.

Remarks

If the supplied OracleDecimal structure has a null value, the returned OracleDecimal has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

SetPrecision

This method returns a new `OracleDecimal` structure with a new specified precision.

Declaration

```
// C#  
public static OracleDecimal SetPrecision(OracleDecimal val, int precision);
```

Parameters

- *val*
An `OracleDecimal` structure.
- *precision*
The specified precision. Range of precision is 1 to 38.

Return Value

An `OracleDecimal` structure.

Remarks

The returned `OracleDecimal` is rounded off if the specified precision is smaller than the precision of *val*.

If *val* has a null value, the returned `OracleDecimal` has a null value.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Types;  
  
class SetPrecisionSample  
{  
    static void Main(string[] args)  
    {  
        OracleDecimal dec1 = new OracleDecimal(555.6666);  
  
        // Set the precision of dec1 to 3  
        OracleDecimal dec2 = OracleDecimal.SetPrecision(dec1, 3);  
  
        // Prints 556  
        Console.WriteLine(dec2.ToString());  
  
        // Set the precision of dec1 to 4  
        OracleDecimal dec3 = OracleDecimal.SetPrecision(dec1, 4);  
  
        // Prints 555.7  
        Console.WriteLine(dec3.ToString());  
    }  
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Shift

This method returns a new `OracleDecimal` structure with its value set to that of the supplied `OracleDecimal` structure, and its decimal place shifted to the specified number of places to the right.

Declaration

```
// C#  
public static OracleDecimal Shift(OracleDecimal val, int decplaces);
```

Parameters

- *val*
An `OracleDecimal` structure.
- *decplaces*
The specified number of places to be shifted.

Return Value

An `OracleDecimal` structure.

Remarks

If the supplied `OracleDecimal` structure has a null value, the returned `OracleDecimal` has a null value.

If *decplaces* is negative, the shift is to the left.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Sign

This method determines the sign of an `OracleDecimal` structure.

Declaration

```
// C#  
public static int Sign(OracleDecimal val);
```

Parameters

- *val*
An OracleDecimal structure.

Return Value

- -1: if the supplied OracleDecimal < 0
- 0: if the supplied OracleDecimal == 0
- 1: if the supplied OracleDecimal > 0

Exceptions

OracleNullValueException - The argument has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Sqrt

This method returns a new OracleDecimal structure with its value set to the square root of the supplied OracleDecimal structure.

Declaration

```
// C#  
public static OracleDecimal Sqrt(OracleDecimal val);
```

Parameters

- *val*
An OracleDecimal structure.

Return Value

An OracleDecimal structure.

Exceptions

ArgumentOutOfRangeException - The provided OracleDecimal structure is less than zero.

Remarks

If either argument has a null value, the returned OracleDecimal has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Subtract

This method returns a new `OracleDecimal` structure with its value set to result of subtracting one `OracleDecimal` structure from another.

Declaration

```
// C#  
public static OracleDecimal Subtract(OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*
An `OracleDecimal` structure.
- *val2*
An `OracleDecimal` structure.

Return Value

An `OracleDecimal` structure.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Truncate

This method truncates the `OracleDecimal` at a specified position.

Declaration

```
// C#  
public static OracleDecimal Truncate(OracleDecimal val, int pos);
```

Parameters

- *val*

An OracleDecimal structure.

- *pos*
The specified position. If the value is positive, the function truncates the OracleDecimal structure to the right of the decimal point. If the value is negative, it truncates the OracleDecimal structure to the left of the decimal point.

Return Value

An OracleDecimal structure.

Remarks

If the supplied OracleDecimal structure has a null value, the returned OracleDecimal has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

OracleDecimal Static (Logarithmic) Methods

The OracleDecimal static (logarithmic) methods are listed in [Table 14-55](#).

Table 14-55 OracleDecimal Static (Logarithmic) Methods

Methods	Description
Exp	Returns a new OracleDecimal structure with its value set to e raised to the supplied power
Log	Returns the supplied OracleDecimal structure with its value set to the logarithm of the supplied OracleDecimal structure (Overloaded)
Pow	Returns a new OracleDecimal structure with its value set to the supplied OracleDecimal structure raised to the supplied power (Overloaded)

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Exp

This method returns a new OracleDecimal structure with its value set to e raised to the supplied OracleDecimal.

Declaration

```
// C#  
public static OracleDecimal Exp(OracleDecimal val);
```

Parameters

- *val*
An OracleDecimal structure.

Return Value

An OracleDecimal structure.

Remarks

If either argument has a null value, the returned OracleDecimal has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Log

Log returns the supplied OracleDecimal structure with its value set to the logarithm of the supplied OracleDecimal structure.

Overload List:

- [Log\(OracleDecimal\)](#)
This method returns a new OracleDecimal structure with its value set to the natural logarithm (base e) of the supplied OracleDecimal structure.
- [Log\(OracleDecimal, int\)](#)
This method returns the supplied OracleDecimal structure with its value set to the logarithm of the supplied OracleDecimal structure in the supplied base.
- [Log\(OracleDecimal, OracleDecimal\)](#)
This method returns the supplied OracleDecimal structure with its value set to the logarithm of the supplied OracleDecimal structure in the supplied base.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Log(OracleDecimal)

This method returns a new `OracleDecimal` structure with its value set to the natural logarithm (base e) of the supplied `OracleDecimal` structure.

Declaration

```
// C#  
public static OracleDecimal Log(OracleDecimal val);
```

Parameters

- `val`
An `OracleDecimal` structure whose logarithm is to be calculated.

Return Value

Returns a new `OracleDecimal` structure with its value set to the natural logarithm (base e) of `val`.

Exceptions

`ArgumentOutOfRangeException` - The supplied `OracleDecimal` value is less than zero.

Remarks

If the supplied `OracleDecimal` structure has a null value, the returned `OracleDecimal` has a null value.

If the supplied `OracleDecimal` structure has zero value, the result is undefined, and the returned `OracleDecimal` structure has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Log(OracleDecimal, int)

This method returns the supplied `OracleDecimal` structure with its value set to the logarithm of the supplied `OracleDecimal` structure in the supplied base.

Declaration

```
// C#  
public static OracleDecimal Log(OracleDecimal val, int logBase);
```

Parameters

- `val`
An `OracleDecimal` structure whose logarithm is to be calculated.

- *logBase*

An int that specifies the base of the logarithm.

Return Value

A new `OracleDecimal` structure with its value set to the logarithm of *val* in the supplied base.

Exceptions

`ArgumentOutOfRangeException` - Either argument is less than zero.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

If both arguments have zero value, the result is undefined, and the returned `OracleDecimal` structure has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Log(OracleDecimal, OracleDecimal)

This method returns the supplied `OracleDecimal` structure with its value set to the logarithm of the supplied `OracleDecimal` structure in the supplied base.

Declaration

```
// C#  
public static OracleDecimal Log(OracleDecimal val, OracleDecimal logBase);
```

Parameters

- *val*
An `OracleDecimal` structure whose logarithm is to be calculated.
- *logBase*
An `OracleDecimal` structure that specifies the base of the logarithm.

Return Value

Returns the logarithm of *val* in the supplied base.

Exceptions

`ArgumentOutOfRangeException` - Either the *val* or *logBase* parameter is less than zero.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

If both arguments have zero value, the result is undefined, and the returned `OracleDecimal` structure has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Pow

`Pow` returns a new `OracleDecimal` structure with its value set to the supplied `OracleDecimal` structure raised to the supplied power.

Overload List:

- [Pow\(OracleDecimal, int\)](#)
This method returns a new `OracleDecimal` structure with its value set to the supplied `OracleDecimal` value raised to the supplied `Int32` power.
- [Pow\(OracleDecimal, OracleDecimal\)](#)
This method returns a new `OracleDecimal` structure with its value set to the supplied `OracleDecimal` structure raised to the supplied `OracleDecimal` power.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Pow(OracleDecimal, int)

This method returns a new `OracleDecimal` structure with its value set to the supplied `OracleDecimal` value raised to the supplied `Int32` power.

Declaration

```
// C#  
public static OracleDecimal Pow(OracleDecimal val, int power);
```

Parameters

- *val*
An `OracleDecimal` structure.
- *power*
An `int` value that specifies the power.

Return Value

An OracleDecimal structure.

Remarks

If the supplied OracleDecimal structure has a null value, the returned OracleDecimal has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Pow(OracleDecimal, OracleDecimal)

This method returns a new OracleDecimal structure with its value set to the supplied OracleDecimal structure raised to the supplied OracleDecimal power.

Declaration

```
// C#  
public static OracleDecimal Pow(OracleDecimal val, OracleDecimal power);
```

Parameters

- *val*
An OracleDecimal structure.
- *power*
An OracleDecimal structure that specifies the power.

Return Value

An OracleDecimal structure.

Remarks

If the supplied OracleDecimal structure has a null value, the returned OracleDecimal has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

OracleDecimal Static (Trigonometric) Methods

The OracleDecimal static (trigonometric) methods are listed in [Table 14-56](#).

Table 14-56 OracleDecimal Static (Trigonometric) Methods

Methods	Description
Acos	Returns an angle in radians whose cosine is the supplied OracleDecimal structure
Asin	Returns an angle in radians whose sine is the supplied OracleDecimal structure
Atan	Returns an angle in radians whose tangent is the supplied OracleDecimal structure
Atan2	Returns an angle in radians whose tangent is the quotient of the two supplied OracleDecimal structures
Cos	Returns the cosine of the supplied angle in radians
Sin	Returns the sine of the supplied angle in radians
Tan	Returns the tangent of the supplied angle in radians
Cosh	Returns the hyperbolic cosine of the supplied angle in radians
Sinh	Returns the hyperbolic sine of the supplied angle in radians
Tanh	Returns the hyperbolic tangent of the supplied angle in radians

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Acos

This method returns an angle in radians whose cosine is the supplied OracleDecimal structure.

Declaration

```
// C#  
public static OracleDecimal Acos(OracleDecimal val);
```

Parameters

- *val*
An OracleDecimal structure. Range is (-1 to 1).

Return Value

An OracleDecimal structure that represents an angle in radians.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Asin

This method returns an angle in radians whose sine is the supplied `OracleDecimal` structure.

Declaration

```
// C#  
public static OracleDecimal Asin(OracleDecimal val);
```

Parameters

- `val`
An `OracleDecimal` structure. Range is (-1 to 1).

Return Value

An `OracleDecimal` structure that represents an angle in radians.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Atan

This method returns an angle in radians whose tangent is the supplied `OracleDecimal` structure.

Declaration

```
// C#  
public static OracleDecimal Atan(OracleDecimal val);
```

Parameters

- *val*
An OracleDecimal.

Return Value

An OracleDecimal structure that represents an angle in radians.

Remarks

If the argument has a null value, the returned OracleDecimal has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Atan2

This method returns an angle in radians whose tangent is the quotient of the two supplied OracleDecimal structures.

Declaration

```
// C#  
public static OracleDecimal Atan2(OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*
An OracleDecimal structure that represents the y-coordinate.
- *val2*
An OracleDecimal structure that represents the x-coordinate.

Return Value

An OracleDecimal structure that represents an angle in radians.

Remarks

If either argument has a null value, the returned OracleDecimal has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Cos

This method returns the cosine of the supplied angle in radians.

Declaration

```
// C#  
public static OracleDecimal Cos(OracleDecimal val);
```

Parameters

- *val*
An OracleDecimal structure that represents an angle in radians.

Return Value

An OracleDecimal instance.

Exceptions

ArgumentOutOfRangeException - The *val* parameter is positive or negative infinity.

Remarks

If either argument has a null value, the returned OracleDecimal has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Sin

This method returns the sine of the supplied angle in radians.

Declaration

```
// C#  
public static OracleDecimal Sin(OracleDecimal val);
```

Parameters

- *val*

An OracleDecimal structure.

Return Value

An OracleDecimal structure that represents an angle in radians.

Exceptions

ArgumentOutOfRangeException - The *val* parameter is positive or negative infinity.

Remarks

If either argument has a null value, the returned OracleDecimal has a null value.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Tan

This method returns the tangent of the supplied angle in radians.

Declaration

```
// C#  
public static OracleDecimal Tan(OracleDecimal val);
```

Parameters

- *val*
An OracleDecimal structure that represents an angle in radians.

Return Value

An OracleDecimal instance.

Exceptions

ArgumentOutOfRangeException - The *val* parameter is positive or negative infinity.

Remarks

If either argument has a null value, the returned OracleDecimal has a null value.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Cosh

This method returns the hyperbolic cosine of the supplied angle in radians.

Declaration

```
// C#  
public static OracleDecimal Cosh(OracleDecimal val);
```

Parameters

- *val*
An OracleDecimal structure that represents an angle in radians.

Return Value

An OracleDecimal instance.

Remarks

If either argument has a null value, the returned OracleDecimal has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Sinh

This method returns the hyperbolic sine of the supplied angle in radians.

Declaration

```
// C#  
public static OracleDecimal Sinh(OracleDecimal val);
```

Parameters

- *val*
An OracleDecimal structure that represents an angle in radians.

Return Value

An OracleDecimal instance.

Remarks

If either argument has a null value, the returned OracleDecimal has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Tanh

This method returns the hyperbolic tangent of the supplied angle in radians.

Declaration

```
// C#
public static OracleDecimal Tanh(OracleDecimal val);
```

Parameters

- `val`
An `OracleDecimal` structure that represents an angle in radians.

Return Value

An `OracleDecimal` instance.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

OracleDecimal Static (Comparison) Operators

The `OracleDecimal` static (comparison) operators are listed in [Table 14-57](#).

Table 14-57 OracleDecimal Static (Comparison) Operators

Operator	Description
operator +	Adds two <code>OracleDecimal</code> values
operator /	Divides one <code>OracleDecimal</code> value by another
operator ==	Determines if the two <code>OracleDecimal</code> values are equal
operator >	Determines if the first of two <code>OracleDecimal</code> values is greater than the second

Table 14-57 (Cont.) OracleDecimal Static (Comparison) Operators

Operator	Description
operator >=	Determines if the first of two OracleDecimal values is greater than or equal to the second
operator !=	Determines if the two OracleDecimal values are not equal
operator <	Determines if the first of two OracleDecimal values is less than the second
operator <=	Determines if the first of two OracleDecimal values is less than or equal to the second
operator *	Multiplies two OracleDecimal structures
operator -	Subtracts one OracleDecimal structure from another
operator -	Negates an OracleDecimal structure
operator%	Returns a new OracleDecimal structure with its value set to the modulus of two OracleDecimal structures.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

operator +

This method adds two OracleDecimal values.

Declaration

```
// C#
public static OracleDecimal operator + (OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*
The first OracleDecimal.
- *val2*
The second OracleDecimal.

Return Value

An OracleDecimal structure.

Remarks

If either operand has a null value, the returned OracleDecimal has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

operator /

This method divides one OracleDecimal value by another.

Declaration

```
/ C#  
public static OracleDecimal operator / (OracleDecimal val1, OracleDecimal val2)
```

Parameters

- *val1*
The first OracleDecimal.
- *val2*
The second OracleDecimal.

Return Value

An OracleDecimal structure.

Remarks

If either operand has a null value, the returned OracleDecimal has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

operator ==

This method determines if two OracleDecimal values are equal.

Declaration

```
// C#  
public static bool operator == (OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*

The first `OracleDecimal`.

- `val2`

The second `OracleDecimal`.

Return Value

Returns `true` if their values are equal; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDecimal` that has a value compares greater than an `OracleDecimal` that has a null value.
- Two `OracleDecimals` that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

operator >

This method determines if the first of two `OracleDecimal` values is greater than the second.

Declaration

```
// C#  
public static bool operator > (OracleDecimal val1, OracleDecimal val2);
```

Parameters

- `val1`

The first `OracleDecimal`.

- `val2`

The second `OracleDecimal`.

Return Value

Returns `true` if the two `OracleDecimal` values are not equal; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDecimal` that has a value compares greater than an `OracleDecimal` that has a null value.
- Two `OracleDecimals` that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

operator >=

This method determines if the first of two `OracleDecimal` values is greater than or equal to the second.

Declaration

```
// C#  
public static bool operator >= (OracleDecimal val1, OracleDecimal val2);
```

Parameters

- `val1`
The first `OracleDecimal`.
- `val2`
The second `OracleDecimal`.

Return Value

Returns `true` if the first of two `OracleDecimal` values is greater than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDecimal` that has a value compares greater than an `OracleDecimal` that has a null value.
- Two `OracleDecimals` that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

operator !=

This method determines if the first of two `OracleDecimal` values are not equal.

Declaration

```
// C#  
public static bool operator != (OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*
The first OracleDecimal.
- *val2*
The second OracleDecimal.

Return Value

Returns true if the two OracleDecimal values are not equal; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleDecimal that has a value compares greater than an OracleDecimal that has a null value.
- Two OracleDecimals that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

operator <

This method determines if the first of two OracleDecimal values is less than the second.

Declaration

```
// C#  
public static bool operator < (OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*
The first OracleDecimal.
- *val2*
The second OracleDecimal.

Return Value

Returns true if the first of two OracleDecimal values is less than the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleDecimal that has a value compares greater than an OracleDecimal that has a null value.
- Two OracleDecimals that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

operator <=

This method determines if the first of two OracleDecimal values is less than or equal to the second.

Declaration

```
// C#  
public static bool operator <= (OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*
The first OracleDecimal.
- *val2*
The second OracleDecimal.

Return Value

Returns true if the first of two OracleDecimal values is less than or equal to the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleDecimal that has a value compares greater than an OracleDecimal that has a null value.
- Two OracleDecimals that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

operator *

This method multiplies two OracleDecimal structures.

Declaration

```
// C#  
public static OracleDecimal operator * (OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*
The first OracleDecimal.
- *val2*
The second OracleDecimal.

Return Value

A new OracleDecimal structure.

Remarks

If either operand has a null value, the returned OracleDecimal has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

operator -

This method subtracts one OracleDecimal structure from another.

Declaration

```
// C#  
public static OracleDecimal operator - (OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*

The first OracleDecimal.

- `val2`

The second OracleDecimal.

Return Value

A new OracleDecimal structure.

Remarks

If either operand has a null value, the returned OracleDecimal has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

operator -

This method negates the supplied OracleDecimal structure.

Declaration

```
// C#  
public static OracleDecimal operator - (OracleDecimal val);
```

Parameters

- `val`
An OracleDecimal.

Return Value

A new OracleDecimal structure.

Remarks

If the supplied OracleDecimal structure has a null value, the returned OracleDecimal has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

operator%

This method returns a new OracleDecimal structure with its value set to the modulus of two OracleDecimal structures.

Declaration

```
// C#
public static OracleDecimal operator % (OracleDecimal val,
    OracleDecimal divider);
```

Parameters

- *val*
An OracleDecimal.
- *divider*
An OracleDecimal.

Return Value

A new OracleDecimal structure.

Remarks

If either operand has a null value, the returned OracleDecimal has a null value.

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

OracleDecimal Static Operators (Conversion from .NET Type to OracleDecimal)

The OracleDecimal static operators (Conversion from .NET Type to OracleDecimal) are listed in [Table 14-58](#).

Table 14-58 OracleDecimal Static Operators (Conversion from .NET Type to OracleDecimal)

Operator	Description
implicit operator OracleDecimal	Converts an instance value to an OracleDecimal structure (Overloaded)
explicit operator OracleDecimal	Converts an instance value to an OracleDecimal structure (Overloaded)

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

implicit operator OracleDecimal

implicit operator OracleDecimal returns the OracleDecimal representation of a value.

Overload List:

- [implicit operator OracleDecimal\(decimal\)](#)
This method returns the OracleDecimal representation of a decimal value.
- [implicit operator OracleDecimal\(int\)](#)
This method returns the OracleDecimal representation of an int value.
- [implicit operator OracleDecimal\(long\)](#)
This method returns the OracleDecimal representation of a long value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

implicit operator OracleDecimal(decimal)

This method returns the OracleDecimal representation of a decimal value.

Declaration

```
// C#  
public static implicit operator OracleDecimal(decimal val);
```

Parameters

- *val*
A decimal value.

Return Value

An OracleDecimal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

implicit operator OracleDecimal(int)

This method returns the `OracleDecimal` representation of an `int` value.

Declaration

```
// C#  
public static implicit operator OracleDecimal(int val);
```

Parameters

- `val`
An `int` value.

Return Value

An `OracleDecimal`.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

implicit operator OracleDecimal(long)

This method returns the `OracleDecimal` representation of a `long` value.

Declaration

```
// C#  
public static implicit operator OracleDecimal(long val);
```

Parameters

- `val`
A `long` value.

Return Value

An `OracleDecimal`.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

explicit operator OracleDecimal

OracleDecimal returns the OracleDecimal representation of a value.

Overload List:

- [explicit operator OracleDecimal\(double\)](#)
This method returns the OracleDecimal representation of a double.
- [explicit operator OracleDecimal\(string\)](#)
This method returns the OracleDecimal representation of a string.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

explicit operator OracleDecimal(double)

This method returns the OracleDecimal representation of a double.

Declaration

```
// C#  
public static explicit operator OracleDecimal(double val);
```

Parameters

- *val*
A double.

Return Value

An OracleDecimal.

Exceptions

OverflowException - The value of the supplied double is greater than the maximum value of OracleDecimal or less than the minimum value of OracleDecimal.

Remarks

OracleDecimal contains the following values depending on the provided double value:

- `double.PositiveInfinity`: positive infinity value
- `double.NegativeInfinity`: negative infinity value.
- `double.NaN`: null value

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

explicit operator OracleDecimal(string)

This method returns the `OracleDecimal` representation of a string.

Declaration

```
// C#  
public static explicit operator OracleDecimal(string numStr);
```

Parameters

- *numStr*
A string that represents a numeric value.

Return Value

An `OracleDecimal`.

Exceptions

`ArgumentException` - The *numStr* parameter is an invalid string representation of an `OracleDecimal`.

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)
- "[OracleGlobalization Class](#)"
- "[Globalization Support](#) "

OracleDecimal Static Operators (Conversion from OracleDecimal to .NET)

The OracleDecimal static operators (Conversion from OracleDecimal to .NET) are listed in [Table 14-59](#).

Table 14-59 OracleDecimal Static Operators (Conversion from OracleDecimal to .NET)

Operator	Description
explicit operator byte	Returns the byte representation of the OracleDecimal value
explicit operator decimal	Returns the decimal representation of the OracleDecimal value
explicit operator double	Returns the double representation of the OracleDecimal value
explicit operator short	Returns the short representation of the OracleDecimal value
explicit operator int	Returns the int representation of the OracleDecimal value
explicit operator long	Returns the long representation of the OracleDecimal value
explicit operator float	Returns the float representation of the OracleDecimal value

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

explicit operator byte

This method returns the byte representation of the OracleDecimal value.

Declaration

```
// C#  
public static explicit operator byte(OracleDecimal val);
```

Parameters

- *val*
An OracleDecimal structure.

Return Value

A byte.

Exceptions

OracleNullValueException - OracleDecimal has a null value.

OverflowException- The byte cannot represent the supplied OracleDecimal structure.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

explicit operator decimal

This method returns the decimal representation of the OracleDecimal value.

Declaration

```
// C#  
public static explicit operator decimal(OracleDecimal val);
```

Parameters

- *val*
An OracleDecimal structure.

Return Value

A decimal.

Exceptions

OracleNullValueException - The OracleDecimal has a null value.

OverflowException - The decimal cannot represent the supplied OracleDecimal structure.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

explicit operator double

This method returns the double representation of the OracleDecimal value.

Declaration

```
// C#  
public static explicit operator double(OracleDecimal val);
```

Parameters

- *val*
An OracleDecimal structure.

Return Value

A double.

Exceptions

OracleNullValueException - The OracleDecimal has a null value.

OverflowException - The double cannot represent the supplied OracleDecimal structure.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

explicit operator short

This method returns the short representation of the OracleDecimal value.

Declaration

```
// C#  
public static explicit operator short(OracleDecimal val);
```

Parameters

- *val*
An OracleDecimal structure.

Return Value

A short.

Exceptions

OracleNullValueException - The OracleDecimal has a null value.

OverflowException - The short cannot represent the supplied OracleDecimal structure.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

explicit operator int

This method returns the int representation of the OracleDecimal value.

Declaration

```
// C#  
public static explicit operator int(OracleDecimal val);
```

Parameters

- *val*
An OracleDecimal structure.

Return Value

An int.

Exceptions

OracleNullValueException - The OracleDecimal has a null value.

OverflowException - The int cannot represent the supplied OracleDecimal structure.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

explicit operator long

This method returns the long representation of the OracleDecimal value.

Declaration

```
// C#  
public static explicit operator long(OracleDecimal val);
```

Parameters

- *val*
An OracleDecimal structure.

Return Value

A long.

Exceptions

OracleNullValueException - The OracleDecimal has a null value.

OverflowException - The long cannot represent the supplied OracleDecimal structure.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

explicit operator float

This method returns the float representation of the OracleDecimal value.

Declaration

```
// C#  
public static explicit operator float(OracleDecimal val);
```

Parameters

- *val*
An OracleDecimal structure.

Return Value

A float.

Exceptions

OracleNullValueException - The OracleDecimal has a null value.

OverflowException - The float cannot represent the supplied OracleDecimal structure.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

OracleDecimal Properties

The OracleDecimal properties are listed in [Table 14-60](#).

Table 14-60 OracleDecimal Properties

Properties	Description
BinData	Returns a byte array that represents the Oracle NUMBER in Oracle internal format
Format	Specifies the format for ToString()
IsInt	Indicates whether or not the current instance is an integer
IsNull	Indicates whether or not the current instance has a null value
IsPositive	Indicates whether or not the current instance is greater than 0
IsZero	Indicates whether or not the current instance has a zero value
Value	Returns a decimal value

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

BinData

This property returns a byte array that represents the Oracle NUMBER in an internal Oracle format.

Declaration

```
// C#  
public byte[] BinData {get;}
```

Property Value

A byte array that represents the Oracle NUMBER in an internal Oracle format.

Exceptions

OracleNullValueException - The current instance has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Format

This property specifies the format for ToString().

Declaration

```
// C#  
public string Format {get; set;}
```

Property Value

The string which specifies the format.

Remarks

Format is used when ToString() is called on an instance of an OracleDecimal. It is useful if the ToString() method needs a specific currency symbol, group, or decimal separator as part of a string.

By default, this property is null which indicates that no special formatting is used.

The decimal and group separator characters are specified by the thread's OracleGlobalization.NumericCharacters.

The currency symbols are specified by the following thread properties:

- OracleGlobalization.Currency
- OracleGlobalization.ISOCurrency
- OracleGlobalization.DualCurrency

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)

IsInt

This property indicates whether or not the current instance is an integer value.

Declaration

```
// C#  
public bool IsInt {get;}
```

Property Value

A bool value that returns true if the current instance is an integer value; otherwise, returns false.

Exceptions

OracleNullValueException - The current instance has a null value.

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

IsNull

This property indicates whether or not the current instance has a null value.

Declaration

```
// C#  
public bool IsNull {get;}
```

Property Value

A bool value that returns true if the current instance has a null value; otherwise, returns false.

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

IsPositive

This property indicates whether or not the value of the current instance is greater than 0.

Declaration

```
// C#  
public bool IsPositive {get;}
```

Property Value

A bool value that returns true if the current instance is greater than 0; otherwise, returns false.

Exceptions

OracleNullValueException - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

IsZero

This property indicates whether or not the current instance has a zero value.

Declaration

```
// C#  
public bool IsZero{get;}
```

Property Value

A bool value that returns true if the current instance has a zero value; otherwise, returns false.

Exceptions

OracleNullValueException - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Value

This method returns a decimal value.

Declaration

```
// C#  
public decimal Value {get;}
```

Property Value

A decimal value.

Exceptions

OracleNullValueException - The current instance has a null value.

OverflowException - The decimal cannot represent the supplied OracleDecimal structure.

Remarks

Precision can be lost when the decimal value is obtained from an `OracleDecimal`. See Remarks under "[OracleDecimal Structure](#)" for further information.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

OracleDecimal Instance Methods

The `OracleDecimal` instance methods are listed in [Table 14-61](#).

Table 14-61 OracleDecimal Instance Methods

Method	Description
CompareTo	Compares the current instance to the supplied object and returns an integer that represents their relative values
Equals	Determines whether or not an object is an instance of <code>OracleDecimal</code> , and whether or not the value of the object is equal to the current instance (Overloaded)
GetHashCode	Returns a hash code for the current instance
GetType	Inherited from <code>System.Object</code>
ToByte	Returns the byte representation of the current instance
ToDouble	Returns the double representation of the current instance
ToInt16	Returns the Int16 representation of the current instance
ToInt32	Returns the Int32 representation of the current instance
ToInt64	Returns the Int64 representation of the current instance
ToSingle	Returns the Single representation of the current instance
ToString	Overloads <code>Object.ToString()</code> Returns the string representation of the current instance

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

CompareTo

This method compares the current instance to the supplied object and returns an integer that represents their relative values.

Declaration

```
// C#  
public int CompareTo(object obj);
```

Parameters

- *obj*
The supplied instance.

Return Value

The method returns a number:

- Less than zero: if the value of the current instance is less than *obj*.
- Zero: if the value of the current instance is equal to *obj*.
- Greater than zero: if the value of the current instance is greater than *obj*.

Implements

IComparable

Exceptions

ArgumentException - The parameter is not of type OracleDecimal.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between OracleDecimals. For example, comparing an OracleDecimal instance with an OracleBinary instance is not allowed. When an OracleDecimal is compared with a different type, an ArgumentException is thrown.
- Any OracleDecimal that has a value compares greater than an OracleDecimal that has a null value.
- Two OracleDecimals that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

Equals

Overrides Object

This method determines whether or not an object is an instance of `OracleDecimal`, and whether or not the value of the object is equal to the current instance.

Declaration

```
// C#  
public override bool Equals(object obj);
```

Parameters

- *obj*
An `OracleDecimal` instance.

Return Value

Returns `true` if *obj* is an instance of `OracleDecimal`, and the value of *obj* is equal to the current instance; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDecimal` that has a value compares greater than an `OracleDecimal` that has a null value.
- Two `OracleDecimals` that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

GetHashCode

Overrides `Object`

This method returns a hash code for the current instance.

Declaration

```
// C#  
public override int GetHashCode();
```

Return Value

Returns a hash code.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

ToByte

This method returns the byte representation of the current instance.

Declaration

```
// C#  
public byte ToByte();
```

Return Value

A byte.

Exceptions

OverflowException - The byte cannot represent the current instance.

OracleNullValueException - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

ToDouble

This method returns the double representation of the current instance.

Declaration

```
// C#  
public double ToDouble();
```

Return Value

A double.

Exceptions

OverflowException - The double cannot represent the current instance.

OracleNullValueException - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

ToInt16

This method returns the Int16 representation of the current instance.

Declaration

```
// C#  
public short ToInt16();
```

Return Value

A short.

Exceptions

OverflowException - The short cannot represent the current instance.

OracleNullValueException - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

ToInt32

This method returns the Int32 representation of the current instance.

Declaration

```
// C#  
public int ToInt32();
```

Return Value

An int.

Exceptions

OverflowException - The int cannot represent the current instance.

OracleNullValueException - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

ToInt64

This method returns the Int64 representation of the current instance.

Declaration

```
// C#  
public long ToInt64();
```

Return Value

A long.

Exceptions

OverflowException - The long cannot represent the current instance.

OracleNullValueException - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

ToSingle

This method returns the Single representation of the current instance.

Declaration

```
// C#  
public float ToSingle();
```

Return Value

A float.

Exceptions

OverflowException - The float cannot represent the current instance.

OracleNullValueException - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)

ToString

Overrides Object

This method returns the string representation of the current instance.

Declaration

```
// C#  
public override string ToString();
```

Return Value

Returns the number in a string returns and a period (.) as a numeric separator.

Remarks

If the current instance has a null value, the returned string is "null".

The returned value is a string representation of an `OracleDecimal` in the numeric format specified by the `Format` property.

The decimal and group separator characters are specified by the thread's `OracleGlobalization.NumericCharacters`.

The currency symbols are specified by the following thread properties:

- `OracleGlobalization.Currency`
- `OracleGlobalization.ISOCurrency`
- `OracleGlobalization.DualCurrency`

If the numeric format is not specified, an Oracle default value is used.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleDecimal Members](#)
- [OracleDecimal Structure](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)

OracleIntervalDS Structure

The OracleIntervalDS structure represents the Oracle INTERVAL DAY TO SECOND data type to be stored in or retrieved from a database. Each OracleIntervalDS stores a period of time in term of days, hours, minutes, seconds, and fractional seconds.

Class Inheritance

```
System.Object
    System.ValueType
        Oracle.DataAccess.Types.OracleIntervalDS
```

Declaration

```
// C#
public struct OracleIntervalDS : IComparable, INullable, IXmlSerializable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#

using System;
using Oracle.DataAccess.Types;

class OracleIntervalDSSample
{
    static void Main()
    {
        OracleIntervalDS iDSMax = OracleIntervalDS.MaxValue;
        double totalDays = iDSMax.TotalDays;

        totalDays -= 1;
        OracleIntervalDS iDSMax_1 = new OracleIntervalDS(totalDays);

        // Calculate the difference
        OracleIntervalDS iDSDiff = iDSMax - iDSMax_1;
```

```
// Prints "idSDiff.ToString() = +0000000000 23:59:59.999999999"  
Console.WriteLine("idSDiff.ToString() = " + idSDiff.ToString());  
}  
}
```

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleIntervalDS Members](#)
- [OracleIntervalDS Constructors](#)
- [OracleIntervalDS Static Fields](#)
- [OracleIntervalDS Static Methods](#)
- [OracleIntervalDS Static Operators](#)
- [OracleIntervalDS Type Conversions](#)
- [OracleIntervalDS Properties](#)
- [OracleIntervalDS Methods](#)

OracleIntervalDS Members

OracleIntervalDS members are listed in the following tables:

OracleIntervalDS Constructors

OracleIntervalDS constructors are listed in [Table 14-62](#)

Table 14-62 OracleIntervalDS Constructors

Constructor	Description
OracleIntervalDS Constructors	Instantiates a new instance of OracleIntervalDS structure (Overloaded)

OracleIntervalDS Static Fields

The OracleIntervalDS static fields are listed in [Table 14-63](#).

Table 14-63 OracleIntervalDS Static Fields

Field	Description
MaxValue	Represents the maximum valid time interval for an OracleIntervalDS structure
MinValue	Represents the minimum valid time interval for an OracleIntervalDS structure
Null	Represents a null value that can be assigned to an OracleIntervalDS instance

Table 14-63 (Cont.) OracleIntervalDS Static Fields

Field	Description
Zero	Represents a zero value for an OracleIntervalDS structure

OracleIntervalDS Static Methods

The OracleIntervalDS static methods are listed in [Table 14-64](#).

Table 14-64 OracleIntervalDS Static Methods

Methods	Description
Equals	Determines whether or not two OracleIntervalDS values are equal (Overloaded)
GreaterThan	Determines whether or not one OracleIntervalDS value is greater than another
GreaterThanOrEqual	Determines whether or not one OracleIntervalDS value is greater than or equal to another
LessThan	Determines whether or not one OracleIntervalDS value is less than another
LessThanOrEqual	Determines whether or not one OracleIntervalDS value is less than or equal to another
NotEquals	Determines whether or not two OracleIntervalDS values are not equal
Parse	Returns an OracleIntervalDS structure and sets its value for time interval using a string
SetPrecision	Returns a new instance of an OracleIntervalDS with the specified day precision and fractional second precision

OracleIntervalDS Static Operators

The OracleIntervalDS static operators are listed in [Table 14-65](#).

Table 14-65 OracleIntervalDS Static Operators

Operator	Description
operator +	Adds two OracleIntervalDS values
operator ==	Determines whether or not two OracleIntervalDS values are equal
operator >	Determines whether or not one OracleIntervalDS value is greater than another
operator >=	Determines whether or not one OracleIntervalDS value is greater than or equal to another
operator !=	Determines whether or not two OracleIntervalDS values are not equal

Table 14-65 (Cont.) OracleIntervalDS Static Operators

Operator	Description
operator <	Determines whether or not one OracleIntervalDS value is less than another
operator <=	Determines whether or not one OracleIntervalDS value is less than or equal to another
operator -	Subtracts one OracleIntervalDS value from another
operator -	Negates an OracleIntervalDS structure
operator *	Multiplies an OracleIntervalDS value by a number
operator /	Divides an OracleIntervalDS value by a number

OracleIntervalDS Type Conversions

The OracleIntervalDS type conversions are listed in [Table 14-66](#).

Table 14-66 OracleIntervalDS Type Conversions

Operator	Description
explicit operator TimeSpan	Converts an OracleIntervalDS structure to a TimeSpan structure
explicit operator OracleIntervalDS	Converts a string to an OracleIntervalDS structure
implicit operator OracleIntervalDS	Converts a TimeSpan structure to an OracleIntervalDS structure

OracleIntervalDS Properties

The OracleIntervalDS properties are listed in [Table 14-67](#).

Table 14-67 OracleIntervalDS Properties

Properties	Description
BinData	Returns an array of bytes that represents the Oracle INTERVAL DAY TO SECOND in Oracle internal format
Days	Gets the days component of an OracleIntervalDS
Hours	Gets the hours component of an OracleIntervalDS
IsNull	Indicates whether or not the current instance has a null value
Milliseconds	Gets the milliseconds component of an OracleIntervalDS
Minutes	Gets the minutes component of an OracleIntervalDS
Nanoseconds	Gets the nanoseconds component of an OracleIntervalDS
Seconds	Gets the seconds component of an OracleIntervalDS
TotalDays	Returns the total number, in days, that represent the time period in the OracleIntervalDS structure

Table 14-67 (Cont.) OracleIntervalDS Properties

Properties	Description
Value	Specifies the time interval that is stored in the OracleIntervalDS structure

OracleIntervalDS Methods

The OracleIntervalDS methods are listed in [Table 14-68](#).

Table 14-68 OracleIntervalDS Methods

Methods	Description
CompareTo	Compares the current OracleIntervalDS instance to an object, and returns an integer that represents their relative values
Equals	Determines whether or not the specified object has the same time interval as the current instance (Overloaded)
GetHashCode	Returns a hash code for the OracleIntervalDS instance
GetType	Inherited from System.Object
ToString	Converts the current OracleIntervalDS structure to a string

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)

OracleIntervalDS Constructors

OracleIntervalDS constructors create a new instance of the OracleIntervalDS structure.

Overload List:

- [OracleIntervalDS\(TimeSpan\)](#)
This constructor creates a new instance of the OracleIntervalDS structure and sets its value using a TimeSpan structure.
- [OracleIntervalDS\(string\)](#)
This constructor creates a new instance of the OracleIntervalDS structure and sets its value using a string that indicates a period of time.
- [OracleIntervalDS\(double\)](#)
This constructor creates a new instance of the OracleIntervalDS structure and sets its value using the total number of days.

- [OracleIntervalDS\(int, int, int, int, double\)](#)

This constructor creates a new instance of the `OracleIntervalDS` structure and sets its value using the supplied days, hours, minutes, seconds and milliseconds.

- [OracleIntervalDS\(int, int, int, int, int\)](#)

This constructor creates a new instance of the `OracleIntervalDS` structure and sets its value using the supplied days, hours, minutes, seconds, and nanoseconds.

- [OracleIntervalDS\(byte\[\]\)](#)

This constructor creates a new instance of the `OracleIntervalDS` structure and sets its value to the provided byte array, which is in an internal Oracle `INTERVAL DAY TO SECOND` format.

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

OracleIntervalDS(TimeSpan)

This constructor creates a new instance of the `OracleIntervalDS` structure and sets its value using a `TimeSpan` structure.

Declaration

```
// C#  
public OracleIntervalDS(TimeSpan ts);
```

Parameters

- *ts*
A `TimeSpan` structure.

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

OracleIntervalDS(string)

This constructor creates a new instance of the `OracleIntervalDS` structure and sets its value using a string that indicates a period of time.

Declaration

```
// C#  
public OracleIntervalDS(string intervalStr);
```

Parameters

- *intervalStr*

A string representing the Oracle INTERVAL DAY TO SECOND.

Exceptions

ArgumentException - The *intervalStr* parameter is not in the valid format or has an invalid value.

ArgumentNullException - The *intervalStr* parameter is null.

Remarks

The value specified in the supplied *intervalStr* must be in Day HH:MI:SSxFF format.

Example

"1 2:3:4.99" means 1 day, 2 hours, 3 minutes, 4 seconds, and 990 milliseconds or 1 day, 2 hours, 3 minutes, 4 seconds, and 990000000 nanoseconds.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

OracleIntervalDS(double)

This constructor creates a new instance of the OracleIntervalDS structure and sets its value using the total number of days.

Declaration

```
// C#  
public OracleIntervalDS(double totalDays);
```

Parameters

- *totalDays*

The supplied total number of days for a time interval. Range of days is $-1000,000,000 < totalDays < 1000,000,000$.

Exceptions

ArgumentOutOfRangeException - The argument value for one or more of the parameters is out of the specified range.

ArgumentException - The argument values of the parameters cannot be used to construct a valid `OracleIntervalDS`.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

OracleIntervalDS(int, int, int, int, double)

This constructor creates a new instance of the `OracleIntervalDS` structure and sets its value using the supplied days, hours, minutes, seconds, and milliseconds.

Declaration

```
// C#
public OracleIntervalDS (int days, int hours, int minutes, int seconds,
    double milliseconds);
```

Parameters

- *days*
The days provided. Range of day is (-999,999,999 to 999,999,999).
- *hours*
The hours provided. Range of hour is (-23 to 23).
- *minutes*
The minutes provided. Range of minute is (-59 to 59).
- *seconds*
The seconds provided. Range of second is (-59 to 59).
- *milliseconds*
The milliseconds provided. Range of millisecond is (- 999.999999 to 999.999999).

Exceptions

ArgumentOutOfRangeException - The argument value for one or more of the parameters is out of the specified range.

ArgumentException - The argument values of the parameters cannot be used to construct a valid `OracleIntervalDS`.

Remarks

The sign of all the arguments must be the same.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

OracleIntervalDS(int, int, int, int, int)

This constructor creates a new instance of the `OracleIntervalDS` structure and sets its value using the supplied days, hours, minutes, seconds, and nanoseconds.

Declaration

```
// C#
public OracleIntervalDS (int days, int hours, int minutes, int seconds,
    int nanoseconds);
```

Parameters

- *days*
The days provided. Range of day is (-999,999,999 to 999,999,999).
- *hours*
The hours provided. Range of hour is (-23 to 23).
- *minutes*
The minutes provided. Range of minute is (-59 to 59).
- *seconds*
The seconds provided. Range of second is (-59 to 59).
- *nanoseconds*
The nanoseconds provided. Range of nanosecond is (-999,999,999 to 999,999,999)

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleIntervalDS`.

Remarks

The sign of all the arguments must be the same.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

OracleIntervalDS(byte[])

This constructor creates a new instance of the OracleIntervalDS structure and sets its value to the provided byte array, which is in an internal Oracle INTERVAL DAY TO SECOND format.

Declaration

```
// C#
public OracleIntervalDS (byte[ ] bytes);
```

Parameters

- *bytes*
A byte array that is in an internal Oracle INTERVAL DAY TO SECOND format.

Exceptions

ArgumentException - bytes is not in internal Oracle INTERVAL DAY TO SECOND format, or bytes is not a valid Oracle INTERVAL DAY TO SECOND.

ArgumentNullException - bytes is null.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

OracleIntervalDS Static Fields

The OracleIntervalDS static fields are listed in [Table 14-69](#).

Table 14-69 OracleIntervalDS Static Fields

Field	Description
MaxValue	Represents the maximum valid time interval for an OracleIntervalDS structure
MinValue	Represents the minimum valid time interval for an OracleIntervalDS structure
Null	Represents a null value that can be assigned to an OracleIntervalDS instance
Zero	Represents a zero value for an OracleIntervalDS structure

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

MaxValue

This static field represents the maximum value for an OracleIntervalDS structure.

Declaration

```
// C#  
public static readonly OracleIntervalDS MaxValue;
```

Remarks

Maximum values:

- Day: 999999999
- hour: 23
- minute is 59
- second: 59
- nanosecond: 999999999

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

MinValue

This static field represents the minimum value for an OracleIntervalDS structure.

Declaration

```
// C#  
public static readonly OracleIntervalDS MinValue;
```

Remarks

Minimum values:

- Day: -999999999
- hour: -23
- minute: -59

- second: -59
- nanosecond: -999999999

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

Null

This static field represents a null value that can be assigned to an `OracleIntervalDS` instance.

Declaration

```
// C#  
public static readonly OracleIntervalDS Null;
```

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

Zero

This static field represents a zero value for an `OracleIntervalDS` structure.

Declaration

```
// C#  
public static readonly OracleIntervalDS Zero;
```

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

OracleIntervalDS Static Methods

The `OracleIntervalDS` static methods are listed in [Table 14-70](#).

Table 14-70 OracleIntervalDS Static Methods

Methods	Description
Equals	Determines whether or not two OracleIntervalDS values are equal (Overloaded)
GreaterThan	Determines whether or not one OracleIntervalDS value is greater than another
GreaterThanOrEqual	Determines whether or not one OracleIntervalDS value is greater than or equal to another
LessThan	Determines whether or not one OracleIntervalDS value is less than another
LessThanOrEqual	Determines whether or not one OracleIntervalDS value is less than or equal to another
NotEquals	Determines whether or not two OracleIntervalDS values are not equal
Parse	Returns an OracleIntervalDS structure and sets its value for time interval using a string
SetPrecision	Returns a new instance of an OracleIntervalDS with the specified day precision and fractional second precision

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

Equals

This static method determines whether or not two OracleIntervalDS values are equal.

Declaration

```
// C#
public static bool Equals(OracleIntervalDS val1, OracleIntervalDS val2);
```

Parameters

- *val1*
The first OracleIntervalDS.
- *val2*
The second OracleIntervalDS.

Return Value

If the two OracleIntervalDS structures represent the same time interval, returns true; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalDS` that has a value compares greater than an `OracleIntervalDS` that has a null value.
- Two `OracleIntervalDS`s that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

GreaterThan

This static method determines whether or not the first of two `OracleIntervalDS` values is greater than the second.

Declaration

```
// C#
public static bool GreaterThan(OracleIntervalDS val1, OracleIntervalDS
    val2);
```

Parameters

- `val1`
The first `OracleIntervalDS`.
- `val2`
The second `OracleIntervalDS`.

Return Value

Returns `true` if the first of two `OracleIntervalDS` values is greater than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalDS` that has a value compares greater than an `OracleIntervalDS` that has a null value.
- Two `OracleIntervalDS`s that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

GreaterThanOrEqual

This static method determines whether or not the first of two `OracleIntervalDS` values is greater than or equal to the second.

Declaration

```
// C#
public static bool GreaterThanOrEqual(OracleIntervalDS val1,
    OracleIntervalDS val2);
```

Parameters

- `val1`
The first `OracleIntervalDS`.
- `val2`
The second `OracleIntervalDS`.

Return Value

Returns `true` if the first of two `OracleIntervalDS` values is greater than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalDS` that has a value compares greater than an `OracleIntervalDS` that has a null value.
- Two `OracleIntervalDS`s that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

LessThan

This static method determines whether or not the first of two `OracleIntervalDS` values is less than the second.

Declaration

```
// C#  
public static bool LessThan(OracleIntervalDS val1, OracleIntervalDS val2);
```

Parameters

- *val1*
The first OracleIntervalDS.
- *val2*
The second OracleIntervalDS.

Return Value

Returns true if the first of two OracleIntervalDS values is less than the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleIntervalDS that has a value compares greater than an OracleIntervalDS that has a null value.
- Two OracleIntervalDSs that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

LessThanOrEqual

This static method determines whether or not the first of two OracleIntervalDS values is less than or equal to the second.

Declaration

```
// C#  
public static bool LessThanOrEqual(OracleIntervalDS val1, OracleIntervalDS val2);
```

Parameters

- *val1*
The first OracleIntervalDS.
- *val2*
The second OracleIntervalDS.

Return Value

Returns true if the first of two `OracleIntervalDS` values is less than or equal to the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalDS` that has a value compares greater than an `OracleIntervalDS` that has a null value.
- Two `OracleIntervalDS`s that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

NotEquals

This static method determines whether or not two `OracleIntervalDS` values are not equal.

Declaration

```
// C#  
public static bool NotEquals(OracleIntervalDS val1, OracleIntervalDS val2);
```

Parameters

- `val1`
The first `OracleIntervalDS`.
- `val2`
The second `OracleIntervalDS`.

Return Value

Returns true if two `OracleIntervalDS` values are not equal; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalDS` that has a value compares greater than an `OracleIntervalDS` that has a null value.
- Two `OracleIntervalDS`s that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

Parse

This static method returns an `OracleIntervalDS` instance and sets its value for time interval using a string.

Declaration

```
// C#  
public static OracleIntervalDS Parse(string intervalStr);
```

Parameters

- *intervalStr*
A string representing the Oracle INTERVAL DAY TO SECOND.

Return Value

Returns an `OracleIntervalDS` instance representing the time interval from the supplied string.

Exceptions

`ArgumentException` - The *intervalStr* parameter is not in the valid format or *intervalStr* has an invalid value.

`ArgumentNullException` - The *intervalStr* parameter is null.

Remarks

The value specified in *intervalStr* must be in Day HH:MI:SSxFF format.

Example

"1 2:3:4.99" means 1 day, 2 hours, 3 minutes, 4 seconds, and 990 milliseconds or 1 day, 2 hours, 3 minutes, 4 seconds, and 990000000 nanoseconds.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

SetPrecision

This static method returns a new instance of an `OracleIntervalDS` with the specified day precision and fractional second precision.

Declaration

```
// C#  
public static OracleIntervalDS SetPrecision(OracleIntervalDS value1,  
    int dayPrecision, int fracSecPrecision);
```

Parameters

- *value1*
An `OracleIntervalDS` structure.
- *dayPrecision*
The day precision provided. Range of day precision is (0 to 9).
- *fracSecPrecision*
The fractional second precision provided. Range of fractional second precision is (0 to 9).

Return Value

An `OracleIntervalDS` instance.

Exceptions

`ArgumentOutOfRangeException` - An argument value is out of the specified range.

Remarks

Depending on the value specified in the supplied *dayPrecision*, 0 or more leading zeros are displayed in the string returned by `ToString()`.

The value specified in the supplied *fracSecPrecision* is used to perform a rounding off operation on the supplied `OracleIntervalDS` value. Depending on this value, 0 or more trailing zeros are displayed in the string returned by `ToString()`.

Example

The `OracleIntervalDS` with a value of "1 2:3:4.99" results in the string "001 2:3:4.99000" when `SetPrecision()` is called, with the day precision set to 3 and fractional second precision set to 5.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

OracleIntervalDS Static Operators

The OracleIntervalDS static operators are listed in [Table 14-71](#).

Table 14-71 OracleIntervalDS Static Operators

Operator	Description
operator +	Adds two OracleIntervalDS values
operator ==	Determines whether or not two OracleIntervalDS values are equal
operator >	Determines whether or not one OracleIntervalDS value is greater than another
operator >=	Determines whether or not one OracleIntervalDS value is greater than or equal to another
operator !=	Determines whether or not two OracleIntervalDS values are not equal
operator <	Determines whether or not one OracleIntervalDS value is less than another
operator <=	Determines whether or not one OracleIntervalDS value is less than or equal to another
operator -	Subtracts one OracleIntervalDS value from another
operator -	Negates an OracleIntervalDS structure
operator *	Multiplies an OracleIntervalDS value by a number
operator /	Divides an OracleIntervalDS value by a number

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

operator +

This static operator adds two OracleIntervalDS values.

Declaration

```
// C#  
public static OracleIntervalDS operator + (OracleIntervalDS val1,  
    OracleIntervalDS val2);
```

Parameters

- *val1*
The first OracleIntervalDS.
- *val2*

The second `OracleIntervalDS`.

Return Value

An `OracleIntervalDS`.

Remarks

If either argument has a null value, the returned `OracleIntervalDS` structure has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

operator ==

This static operator determines if two `OracleIntervalDS` values are equal.

Declaration

```
// C#  
public static bool operator == (OracleIntervalDS val1,  
    OracleIntervalDS val2);
```

Parameters

- *val1*
The first `OracleIntervalDS`.
- *val2*
The second `OracleIntervalDS`.

Return Value

Returns true if the two `OracleIntervalDS` values are the same; otherwise returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalDS` that has a value compares greater than an `OracleIntervalDS` that has a null value.
- Two `OracleIntervalDS`s that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

operator >

This static operator determines if the first of two `OracleIntervalDS` values is greater than the second.

Declaration

```
// C#  
public static bool operator > (OracleIntervalDS val1,  
    OracleIntervalDS val2);
```

Parameters

- `val1`
The first `OracleIntervalDS`.
- `val2`
The second `OracleIntervalDS`.

Return Value

Returns true if one `OracleIntervalDS` value is greater than another; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalDS` that has a value compares greater than an `OracleIntervalDS` that has a null value.
- Two `OracleIntervalDS`s that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

operator >=

This static operator determines if the first of two `OracleIntervalDS` values is greater than or equal to the second.

Declaration

```
// C#  
public static bool operator >= (OracleIntervalDS val1,  
    OracleIntervalDS val2);
```

Parameters

- *val1*
The first OracleIntervalDS.
- *val2*
The second OracleIntervalDS.

Return Value

Returns true if the first of two OracleIntervalDS values is greater than or equal to the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleIntervalDS that has a value compares greater than an OracleIntervalDS that has a null value.
- Two OracleIntervalDSs that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

operator !=

This static operator determines if the two OracleIntervalDS values are not equal.

Declaration

```
// C#  
public static bool operator != (OracleIntervalDS val1,  
    OracleIntervalDS val2);
```

Parameters

- *val1*
The first OracleIntervalDS.
- *val2*
The second OracleIntervalDS.

Return Value

Returns true if the two OracleIntervalDS values are not equal; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleIntervalDS that has a value compares greater than an OracleIntervalDS that has a null value.
- Two OracleIntervalDSs that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

operator <

This static operator determines if the first of two OracleIntervalDS values is less than the second.

Declaration

```
// C#  
public static bool operator < (OracleIntervalDS val1,  
    OracleIntervalDS val2);
```

Parameters

- *val1*
The first OracleIntervalDS.
- *val2*
The second OracleIntervalDS.

Return Value

Returns true if the first of two OracleIntervalDS values is less than the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleIntervalDS that has a value compares greater than an OracleIntervalDS that has a null value.
- Two OracleIntervalDSs that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

operator <=

This static operator determines if the first of two `OracleIntervalDS` values is less than or equal to the second.

Declaration

```
// C#  
public static bool operator <= (OracleIntervalDS val1,  
    OracleIntervalDS val2);
```

Parameters

- `val1`
The first `OracleIntervalDS`.
- `val2`
The second `OracleIntervalDS`.

Return Value

Returns `true` if the first of two `OracleIntervalDS` values is less than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalDS` that has a value compares greater than an `OracleIntervalDS` that has a null value.
- Two `OracleIntervalDS`s that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

operator -

This static operator subtracts one `OracleIntervalDS` structure from another.

Declaration

```
// C#  
public static OracleIntervalDS operator - (OracleIntervalDS val1,  
    OracleIntervalDS val2);
```

Parameters

- *val1*
The first OracleIntervalDS.
- *val2*
The second OracleIntervalDS.

Return Value

An OracleIntervalDS structure.

Remarks

If either argument has a null value, the returned OracleIntervalDS structure has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

operator -

This static operator negates the supplied OracleIntervalDS structure.

Declaration

```
// C#  
public static OracleIntervalDS operator - (OracleIntervalDS val);
```

Parameters

- *val*
An OracleIntervalDS.

Return Value

An OracleIntervalDS structure.

Remarks

If the supplied OracleIntervalDS structure has a null value, the returned OracleIntervalDS structure has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

operator *

This static operator multiplies an `OracleIntervalDS` value by a number.

Declaration

```
// C#  
public static OracleIntervalDS operator * (OracleIntervalDS val1,  
    int multiplier);
```

Parameters

- *val1*
The first `OracleIntervalDS`.
- *multiplier*
A multiplier.

Return Value

A new `OracleIntervalDS` instance.

Remarks

If the `OracleIntervalDS` structure has a null value, the returned `OracleIntervalDS` structure has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

operator /

This static operator divides an `OracleIntervalDS` value by a number.

Declaration

```
// C#  
public static OracleIntervalDS operator / (OracleIntervalDS val1,  
    int divisor);
```

Parameters

- val1*
The first OracleIntervalDS.
- divisor*
A divisor.

Return Value

An OracleIntervalDS structure.

Remarks

If the OracleIntervalDS structure has a null value, the returned OracleIntervalDS structure has a null value.

❶ See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

OracleIntervalDS Type Conversions

The OracleIntervalDS type conversions are listed in [Table 14-72](#).

Table 14-72 OracleIntervalDS Type Conversions

Operator	Description
explicit operator TimeSpan	Converts an OracleIntervalDS structure to a TimeSpan structure
explicit operator OracleIntervalDS	Converts a string to an OracleIntervalDS structure
implicit operator OracleIntervalDS	Converts a TimeSpan structure to an OracleIntervalDS structure

❶ See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

explicit operator TimeSpan

This type conversion operator converts an `OracleIntervalDS` structure to a `TimeSpan` structure.

Declaration

```
// C#  
public static explicit operator TimeSpan(OracleIntervalDS val);
```

Parameters

- *val*
An `OracleIntervalDS` instance.

Return Value

A `TimeSpan` structure.

Exceptions

`OracleNullValueException` - The `OracleIntervalDS` structure has a null value.

Remarks

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

explicit operator OracleIntervalDS

This type conversion operator converts a string to an `OracleIntervalDS` structure.

Declaration

```
// C#  
public static explicit operator OracleIntervalDS (string intervalStr);
```

Parameters

- *intervalStr*
A string representation of an Oracle INTERVAL DAY TO SECOND.

Return Value

An `OracleIntervalDS` structure.

Exceptions

`ArgumentException` - The supplied *intervalStr* parameter is not in the correct format or has an invalid value.

ArgumentNullException - The *intervalStr* parameter is null.

Remarks

The returned OracleIntervalDS structure contains the same time interval represented by the supplied *intervalStr*. The value specified in the supplied *intervalStr* must be in Day HH:MI:SSxFF format.

Example

"1 2:3:4.99" means 1 day, 2 hours, 3 minutes 4 seconds and 990 milliseconds or 1 day, 2 hours, 3 minutes 4 seconds and 990000000 nanoseconds.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

implicit operator OracleIntervalDS

This type conversion operator converts a TimeSpan structure to an OracleIntervalDS structure.

Declaration

```
// C#  
public static implicit operator OracleIntervalDS(TimeSpan val);
```

Parameters

- *val*
A TimeSpan instance.

Return Value

An OracleIntervalDS structure.

Remarks

The returned OracleIntervalDS structure contains the same days, hours, seconds, and milliseconds as the supplied TimeSpan *val*.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

OracleIntervalDS Properties

The OracleIntervalDS properties are listed in [Table 14-73](#).

Table 14-73 OracleIntervalDS Properties

Properties	Description
BinData	Returns an array of bytes that represents the Oracle INTERVAL DAY TO SECOND in Oracle internal format
Days	Gets the days component of an OracleIntervalDS
Hours	Gets the hours component of an OracleIntervalDS
IsNull	Indicates whether or not the current instance has a null value
Milliseconds	Gets the milliseconds component of an OracleIntervalDS
Minutes	Gets the minutes component of an OracleIntervalDS
Nanoseconds	Gets the nanoseconds component of an OracleIntervalDS
Seconds	Gets the seconds component of an OracleIntervalDS
TotalDays	Returns the total number, in days, that represent the time period in the OracleIntervalDS structure
Value	Specifies the time interval that is stored in the OracleIntervalDS structure

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

BinData

This property returns an array of bytes that represents the Oracle INTERVAL DAY TO SECOND in Oracle internal format.

Declaration

```
// C#  
public byte[] BinData {get;}
```

Property Value

A byte array that represents an Oracle INTERVAL DAY TO SECOND in Oracle internal format.

Exceptions

OracleNullValueException - The current instance has a null value.

Remarks

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

Days

This property gets the days component of an `OracleIntervalDS`.

Declaration

```
// C#  
public int Days {get;}
```

Property Value

An `int` representing the days component.

Exceptions

`OracleNullValueException` - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

Hours

This property gets the hours component of an `OracleIntervalDS`.

Declaration

```
// C#  
public int Hours {get;}
```

Property Value

An `int` representing the hours component.

Exceptions

`OracleNullValueException` - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

IsNull

This property indicates whether or not the current instance has a null value.

Declaration

```
// C#  
public bool IsNull {get;}
```

Property Value

Returns true if the current instance has a null value; otherwise, returns false.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

Milliseconds

This property gets the milliseconds component of an OracleIntervalDS.

Declaration

```
// C#  
public double Milliseconds {get;}
```

Property Value

A double that represents milliseconds component.

Exceptions

OracleNullValueException - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

Minutes

This property gets the minutes component of an `OracleIntervalDS`.

Declaration

```
// C#  
public int Minutes {get;}
```

Property Value

A `int` that represents minutes component.

Exceptions

`OracleNullValueException` - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

Nanoseconds

This property gets the nanoseconds component of an `OracleIntervalDS`.

Declaration

```
// C#  
public int Nanoseconds {get;}
```

Property Value

An `int` that represents nanoseconds component.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

Seconds

This property gets the seconds component of an `OracleIntervalDS`.

Declaration

```
// C#  
public int Seconds {get;}
```

Property Value

An `int` that represents seconds component.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

TotalDays

This property returns the total number, in days, that represent the time period in the `OracleIntervalDS` structure.

Declaration

```
// C#  
public double TotalDays {get;}
```

Property Value

A `double` that represents the total number of days.

Exceptions

`OracleNullValueException` - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

Value

This property specifies the time interval that is stored in the OracleIntervalDS structure.

Declaration

```
// C#
public TimeSpan Value {get;}
```

Property Value

A time interval.

Exceptions

OracleNullValueException - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

OracleIntervalDS Methods

The OracleIntervalDS methods are listed in [Table 14-74](#).

Table 14-74 OracleIntervalDS Methods

Methods	Description
CompareTo	Compares the current OracleIntervalDS instance to an object, and returns an integer that represents their relative values
Equals	Determines whether or not the specified object has the same time interval as the current instance (Overloaded)
GetHashCode	Returns a hash code for the OracleIntervalDS instance
GetType	Inherited from System.Object
ToString	Converts the current OracleIntervalDS structure to a string

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

CompareTo

This method compares the current `OracleIntervalDS` instance to an object, and returns an integer that represents their relative values.

Declaration

```
// C#  
public int CompareTo(object obj);
```

Parameters

- *obj*
The object being compared to.

Return Value

The method returns:

- Less than zero: if the current `OracleIntervalDS` represents a shorter time interval than *obj*.
- Zero: if the current `OracleIntervalDS` and *obj* represent the same time interval.
- Greater than zero: if the current `OracleIntervalDS` represents a longer time interval than *obj*.

Implements

`IComparable`

Exceptions

`ArgumentException` - The *obj* parameter is not of type `OracleIntervalDS`.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between `OracleIntervalDS`s. For example, comparing an `OracleIntervalDS` instance with an `OracleBinary` instance is not allowed. When an `OracleIntervalDS` is compared with a different type, an `ArgumentException` is thrown.
- Any `OracleIntervalDS` that has a value compares greater than an `OracleIntervalDS` that has a null value.
- Two `OracleIntervalDS`s that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

Equals

This method determines whether or not the specified object has the same time interval as the current instance.

Declaration

```
// C#  
public override bool Equals(object obj);
```

Parameters

- *obj*
The specified object.

Return Value

Returns true if *obj* is of type `OracleIntervalDS` and has the same time interval as the current instance; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalDS` that has a value compares greater than an `OracleIntervalDS` that has a null value.
- Two `OracleIntervalDS`s that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

GetHashCode

Overrides `Object`

This method returns a hash code for the `OracleIntervalDS` instance.

Declaration

```
// C#  
public override int GetHashCode();
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

ToString

Overrides Object

This method converts the current OracleIntervalDS structure to a string.

Declaration

```
// C#  
public override string ToString();
```

Return Value

Returns a string.

Remarks

If the current instance has a null value, the returned string contains "null".

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalDS Structure](#)
- [OracleIntervalDS Members](#)

OracleIntervalYM Structure

The OracleIntervalYM structure represents the Oracle INTERVAL YEAR TO MONTH data type to be stored in or retrieved from a database. Each OracleIntervalYM stores a period of time in years and months.

Class Inheritance

System.Object

System.ValueType

Oracle.DataAccess.Types.OracleIntervalYM

Declaration

```
// C#  
public struct OracleIntervalYM : IComparable
```


Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#

using System;
using Oracle.DataAccess.Types;

class OracleIntervalYMSample
{
    static void Main()
    {
        OracleIntervalYM iYMMax = OracleIntervalYM.MaxValue;
        double totalYears = iYMMax.TotalYears;

        totalYears -= 1;
        OracleIntervalYM iYMMax_1 = new OracleIntervalYM(totalYears);

        // Calculate the difference
        OracleIntervalYM iYMDiff = iYMMax - iYMMax_1;

        // Prints "iYMDiff.ToString() = +000000001-00"
        Console.WriteLine("iYMDiff.ToString() = " + iYMDiff.ToString());
    }
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Members](#)
- [OracleIntervalYM Constructors](#)
- [OracleIntervalYM Static Fields](#)
- [OracleIntervalYM Static Methods](#)
- [OracleIntervalYM Static Operators](#)
- [OracleIntervalYM Type Conversions](#)
- [OracleIntervalYM Properties](#)
- [OracleIntervalYM Methods](#)

OracleIntervalYM Members

OracleIntervalYM members are listed in the following tables:

OracleIntervalYM Constructors

OracleIntervalYM constructors are listed in [Table 14-75](#)

Table 14-75 OracleIntervalYM Constructors

Constructor	Description
OracleIntervalYM Constructors	Instantiates a new instance of OracleIntervalYM structure (Overloaded)

OracleIntervalYM Static Fields

The OracleIntervalYM static fields are listed in [Table 14-76](#).

Table 14-76 OracleIntervalYM Static Fields

Field	Description
MaxValue	Represents the maximum value for an OracleIntervalYM structure
MinValue	Represents the minimum value for an OracleIntervalYM structure
Null	Represents a null value that can be assigned to an OracleIntervalYM instance
Zero	Represents a zero value for an OracleIntervalYM structure

OracleIntervalYM Static Methods

The OracleIntervalYM static methods are listed in [Table 14-77](#).

Table 14-77 OracleIntervalYM Static Methods

Methods	Description
Equals	Determines whether or not two OracleIntervalYM values are equal (Overloaded)
GreaterThan	Determines whether or not one OracleIntervalYM value is greater than another
GreaterThanOrEqual	Determines whether or not one OracleIntervalYM value is greater than or equal to another
LessThan	Determines whether or not one OracleIntervalYM value is less than another
LessThanOrEqual	Determines whether or not one OracleIntervalYM value is less than or equal to another
NotEquals	Determines whether two OracleIntervalYM values are not equal
Parse	Returns an OracleIntervalYM structure and sets its value for time interval using a string
SetPrecision	Returns a new instance of an OracleIntervalYM with the specified year precision.

OracleIntervalYM Static Operators

The OracleIntervalYM static operators are listed in [Table 14-78](#).

Table 14-78 OracleIntervalYM Static Operators

Operator	Description
operator +	Adds two OracleIntervalYM values
operator ==	Determines whether or not two OracleIntervalYM values are equal
operator >	Determines whether or not one OracleIntervalYM value is greater than another
operator >=	Determines whether or not one OracleIntervalYM value is greater than or equal to another
operator !=	Determines whether two OracleIntervalYM values are not equal
operator <	Determines whether or not one OracleIntervalYM value is less than another
operator <=	Determines whether or not one OracleIntervalYM value is less than or equal to another
operator -	Subtracts one OracleIntervalYM value from another
operator -	Negates an OracleIntervalYM structure
operator *	Multiplies an OracleIntervalYM value by a number
operator /	Divides an OracleIntervalYM value by a number

OracleIntervalYM Type Conversions

The OracleIntervalYM conversions are listed in [Table 14-79](#).

Table 14-79 OracleIntervalYM Type Conversions

Operator	Description
explicit operator long	Converts an OracleIntervalYM structure to a number
explicit operator OracleIntervalYM	Converts a string to an OracleIntervalYM structure
implicit operator OracleIntervalYM	Converts the number of months to an OracleIntervalYM structure

OracleIntervalYM Properties

The OracleIntervalYM properties are listed in [Table 14-80](#).

Table 14-80 OracleIntervalYM Properties

Properties	Description
BinData	Returns an array of bytes that represents the Oracle INTERVAL YEAR TO MONTH in an Oracle internal format
IsNull	Indicates whether or not the current instance has a null value
Months	Gets the months component of an OracleIntervalYM
TotalYears	Returns the total number, in years, that represents the period of time in the current OracleIntervalYM structure
Value	Specifies the total number of months that is stored in the OracleIntervalYM structure
Years	Gets the years component of an OracleIntervalYM

OracleIntervalYM Methods

The OracleIntervalYM methods are listed in [Table 14-81](#).

Table 14-81 OracleIntervalYM Methods

Methods	Description
CompareTo	Compares the current OracleIntervalYM instance to the supplied object, and returns an integer that represents their relative values
Equals	Determines whether or not the specified object has the same time interval as the current instance (Overloaded)
GetHashCode	Returns a hash code for the OracleIntervalYM instance
GetType	Inherited from System.Object
ToString	Converts the current OracleIntervalYM structure to a string

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)

OracleIntervalYM Constructors

The OracleIntervalYM constructors creates a new instance of the OracleIntervalYM structure.

Overload List:

- [OracleIntervalYM\(long\)](#)
This method creates a new instance of the OracleIntervalYM structure using the supplied total number of months for a period of time.
- [OracleIntervalYM\(string\)](#)
This method creates a new instance of the OracleIntervalYM structure and sets its value using the supplied string.
- [OracleIntervalYM\(double\)](#)
This method creates a new instance of the OracleIntervalYM structure and sets its value using the total number of years.
- [OracleIntervalYM\(int, int\)](#)
This method creates a new instance of the OracleIntervalYM structure and sets its value using years and months.
- [OracleIntervalYM\(byte\[\]\)](#)
This method creates a new instance of the OracleIntervalYM structure and sets its value to the provided byte array, which is in an internal Oracle INTERVAL DAY TO SECOND format.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

OracleIntervalYM(long)

This method creates a new instance of the OracleIntervalYM structure using the supplied total number of months for a period of time.

Declaration

```
// C#  
public OracleIntervalYM (long totalMonths);
```

Parameters

- *totalMonths*

The number of total months for a time interval. Range is $-12,000,000,000 < totalMonths < 12,000,000,000$.

Exceptions

ArgumentOutOfRangeException - The *totalMonths* parameter is out of the specified range.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

OracleIntervalYM(string)

This method creates a new instance of the OracleIntervalYM structure and sets its value using the supplied string.

Declaration

```
// C#  
public OracleIntervalYM (string intervalStr);
```

Parameters

- *intervalStr*

A string representing the Oracle INTERVAL YEAR TO MONTH.

Remarks

The value specified in the supplied *intervalStr* must be in Year-Month format.

Exceptions

ArgumentException - The *intervalStr* parameter is not in the valid format or *intervalStr* has an invalid value.

ArgumentNullException - The *intervalStr* parameter is null.

Example

"1-2" means 1 year and 2 months.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

OracleIntervalYM(double)

This method creates a new instance of the `OracleIntervalYM` structure and sets its value using the total number of years.

Declaration

```
// C#  
public OracleIntervalYM (double totalYears);
```

Parameters

- *totalYears*
Number of total years. Range is $-1,000,000,000 < totalYears < 1,000,000,000$.

Exceptions

`ArgumentOutOfRangeException` - The *totalYears* parameter is out of the specified range.

`ArgumentException` - The *totalYears* parameter cannot be used to construct a valid `OracleIntervalYM`.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

OracleIntervalYM(int, int)

This method creates a new instance of the `OracleIntervalYM` structure and sets its value using years and months.

Declaration

```
// C#  
public OracleIntervalYM (int years, int months);
```

Parameters

- *years*
Number of years. Range of year is $(-999,999,999 \text{ to } 999,999,999)$.
- *months*

Number of months. Range of month is (-11 to 11).

Remarks

The sign of all the arguments must be the same.

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleIntervalYM`.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

OracleIntervalYM(byte[])

This method creates a new instance of the `OracleIntervalYM` structure and sets its value to the provided byte array, which is in an internal Oracle INTERVAL DAY TO SECOND format.

Declaration

```
// C#  
public OracleIntervalYM (byte[] bytes);
```

Parameters

- *bytes*
A byte array that is in an internal Oracle INTERVAL YEAR TO MONTH format.

Exceptions

`ArgumentException` - The supplied byte array is not in an internal Oracle INTERVAL YEAR TO MONTH format or the supplied byte array has an invalid value.

`ArgumentNullException` - *bytes* is null.

Remarks

The supplied byte array must be in an internal Oracle INTERVAL YEAR TO MONTH format.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

OracleIntervalYM Static Fields

The OracleIntervalYM static fields are listed in [Table 14-82](#).

Table 14-82 OracleIntervalYM Static Fields

Field	Description
MaxValue	Represents the maximum value for an OracleIntervalYM structure
MinValue	Represents the minimum value for an OracleIntervalYM structure
Null	Represents a null value that can be assigned to an OracleIntervalYM instance
Zero	Represents a zero value for an OracleIntervalYM structure

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

MaxValue

This static field represents the maximum value for an OracleIntervalYM structure.

Declaration

```
// C#  
public static readonly OracleIntervalYM MaxValue;
```

Remarks

Year is 999999999 and Month is 11.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

MinValue

This static field represents the minimum value for an OracleIntervalYM structure.

Declaration

```
// C#  
public static readonly OracleIntervalYM MinValue;
```

Remarks

Year is -999999999 and Month is -11.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

Null

This static field represents a null value that can be assigned to an OracleIntervalYM instance.

Declaration

```
// C#  
public static readonly OracleIntervalYM Null;
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

Zero

This static field represents a zero value for an OracleIntervalYM structure.

Declaration

```
// C#  
public static readonly OracleIntervalDS Zero;
```

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

OracleIntervalYM Static Methods

The OracleIntervalYM static methods are listed in [Table 14-83](#).

Table 14-83 OracleIntervalYM Static Methods

Methods	Description
Equals	Determines whether or not two OracleIntervalYM values are equal (Overloaded)
GreaterThan	Determines whether or not one OracleIntervalYM value is greater than another
GreaterThanOrEqual	Determines whether or not one OracleIntervalYM value is greater than or equal to another
LessThan	Determines whether or not one OracleIntervalYM value is less than another
LessThanOrEqual	Determines whether or not one OracleIntervalYM value is less than or equal to another
NotEquals	Determines whether two OracleIntervalYM values are not equal
Parse	Returns an OracleIntervalYM structure and sets its value for time interval using a string
SetPrecision	Returns a new instance of an OracleIntervalYM with the specified year precision.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

Equals

This static method determines whether or not two OracleIntervalYM values are equal.

Declaration

```
// C#  
public static bool Equals(OracleIntervalYM val1, OracleIntervalYM val2);
```

Parameters

- *val1*
An OracleIntervalYM structure.
- *val2*
An OracleIntervalYM structure.

Return Value

Returns true if two OracleIntervalYM values represent the same time interval, otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleIntervalYM that has a value compares greater than an OracleIntervalYM that has a null value.
- Two OracleIntervalYMs that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

GreaterThan

This static method determines whether or not the first of two OracleIntervalYM values is greater than the second.

Declaration

```
// C#  
public static bool GreaterThan(OracleIntervalYM val1, OracleIntervalYM val2);
```

Parameters

- *val1*
The first OracleIntervalYM.
- *val2*
The second OracleIntervalYM.

Return Value

Returns true if the first of two OracleIntervalYM values is greater than the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalYM` that has a value compares greater than an `OracleIntervalYM` that has a null value.
- Two `OracleIntervalYMs` that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

GreaterThanOrEqual

This static method determines whether or not the first of two `OracleIntervalYM` values is greater than or equal to the second.

Declaration

```
// C#
public static bool GreaterThanOrEqual(OracleIntervalYM val1,
    OracleIntervalYM val2);
```

Parameters

- `val1`
The first `OracleIntervalYM`.
- `val2`
The second `OracleIntervalYM`.

Return Value

Returns `true` if the first of two `OracleIntervalYM` values is greater than or equal to the second; otherwise returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalYM` that has a value compares greater than an `OracleIntervalYM` that has a null value.
- Two `OracleIntervalYMs` that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

LessThan

This static method determines whether or not the first of two `OracleIntervalYM` values is less than the second.

Declaration

```
// C#  
public static bool LessThan(OracleIntervalYM val1, OracleIntervalYM val2);
```

Parameters

- *val1*
The first `OracleIntervalYM`.
- *val2*
The second `OracleIntervalYM`.

Return Value

Returns `true` if the first of two `OracleIntervalYM` values is less than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalYM` that has a value compares greater than an `OracleIntervalYM` that has a null value.
- Two `OracleIntervalYMs` that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

LessThanOrEqual

This static method determines whether or not the first of two `OracleIntervalYM` values is less than or equal to the second.

Declaration

```
// C#  
public static bool LessThanOrEqual(OracleIntervalYM val1, OracleIntervalYM val2);
```

Parameters

- *val1*
The first OracleIntervalYM.
- *val2*
The second OracleIntervalYM.

Return Value

Returns true if the first of two OracleIntervalYM values is less than or equal to the second. Returns false otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any OracleIntervalYM that has a value compares greater than an OracleIntervalYM that has a null value.
- Two OracleIntervalYMs that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

NotEquals

This static method determines whether two OracleIntervalYM values are not equal.

Declaration

```
// C#  
public static bool NotEquals(OracleIntervalYM val1, OracleIntervalYM val2);
```

Parameters

- *val1*
The first OracleIntervalYM.
- *val2*
The second OracleIntervalYM.

Return Value

Returns true if two OracleIntervalYM values are not equal. Returns false otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any OracleIntervalYM that has a value compares greater than an OracleIntervalYM that has a null value.
- Two OracleIntervalYMs that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

Parse

This static method returns an OracleIntervalYM structure and sets its value for time interval using a string.

Declaration

```
// C#  
public static OracleIntervalYM Parse (string intervalStr);
```

Parameters

- *intervalStr*
A string representing the Oracle INTERVAL YEAR TO MONTH.

Return Value

Returns an OracleIntervalYM structure.

Exceptions

ArgumentException - The *intervalStr* parameter is not in the valid format or *intervalStr* has an invalid value.

ArgumentNullException - The *intervalStr* parameter is null.

Remarks

The value specified in the supplied *intervalStr* must be in the Year-Month format.

Example

"1-2" means 1 year and 2 months.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

SetPrecision

This static method returns a new instance of an `OracleIntervalYM` with the specified year precision.

Declaration

```
// C#
public static OracleIntervalYM SetPrecision(OracleIntervalYM value1,
    int yearPrecision);
```

Parameters

- *value1*
An `OracleIntervalYM` structure.
- *yearPrecision*
The year precision provided. Range of year precision is (0 to 9).

Return Value

An `OracleIntervalDS` instance.

Exceptions

`ArgumentOutOfRangeException` - *yearPrecision* is out of the specified range.

Remarks

Depending on the value specified in the supplied *yearPrecision*, 0 or more leading zeros are displayed in the string returned by `ToString()`.

Example

An `OracleIntervalYM` with a value of "1-2" results in the string "001-2" when `SetPrecision()` is called with the year precision set to 3.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

OracleIntervalYM Static Operators

The OracleIntervalYM static operators are listed in [Table 14-84](#).

Table 14-84 OracleIntervalYM Static Operators

Operator	Description
operator +	Adds two OracleIntervalYM values
operator ==	Determines whether or not two OracleIntervalYM values are equal
operator >	Determines whether or not one OracleIntervalYM value is greater than another
operator >=	Determines whether or not one OracleIntervalYM value is greater than or equal to another
operator !=	Determines whether two OracleIntervalYM values are not equal
operator <	Determines whether or not one OracleIntervalYM value is less than another
operator <=	Determines whether or not one OracleIntervalYM value is less than or equal to another
operator -	Subtracts one OracleIntervalYM value from another
operator -	Negates an OracleIntervalYM structure
operator *	Multiplies an OracleIntervalYM value by a number
operator /	Divides an OracleIntervalYM value by a number

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

operator +

This static operator adds two OracleIntervalYM values.

Declaration

```
// C#
public static OracleIntervalYM operator + (OracleIntervalYM val1,
    OracleIntervalYM val2);
```

Parameters

- *val1*
The first OracleIntervalYM.
- *val2*
The second OracleIntervalYM.

Return Value

OracleIntervalYM

Remarks

If either argument has a null value, the returned OracleIntervalYM structure has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

operator ==

This static operator determines if two OracleIntervalYM values are equal.

Declaration

```
// C#  
public static bool operator == (OracleIntervalYM val1, OracleIntervalYM val2);
```

Parameters

- *val1*
The first OracleIntervalYM.
- *val2*
The second OracleIntervalYM.

Return Value

Returns true if they are equal; otherwise returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleIntervalYM that has a value compares greater than an OracleIntervalYM that has a null value.
- Two OracleIntervalYMs that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

operator >

This static operator determines if the first of two `OracleIntervalYM` values is greater than the second.

Declaration

```
// C#  
public static bool operator > (OracleIntervalYM val1, OracleIntervalYM val2);
```

Parameters

- *val1*
The first `OracleIntervalYM`.
- *val2*
The second `OracleIntervalYM`.

Return Value

Returns true if one `OracleIntervalYM` value is greater than another; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalYM` that has a value compares greater than an `OracleIntervalYM` that has a null value.
- Two `OracleIntervalYMs` that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

operator >=

This static operator determines if the first of two `OracleIntervalYM` values is greater than or equal to the second.

Declaration

```
// C#  
public static bool operator >= (OracleIntervalYM val1, OracleIntervalYM val2);
```

Parameters

- *val1*
The first `OracleIntervalYM`.
- *val2*

The second `OracleIntervalYM`.

Return Value

Returns true if one `OracleIntervalYM` value is greater than or equal to another; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalYM` that has a value compares greater than an `OracleIntervalYM` that has a null value.
- Two `OracleIntervalYMs` that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

operator !=

This static operator determines whether two `OracleIntervalYM` values are not equal.

Declaration

```
// C#  
public static bool operator != (OracleIntervalYM val1, OracleIntervalYM val2)
```

Parameters

- `val1`
The first `OracleIntervalYM`.
- `val2`
The second `OracleIntervalYM`.

Return Value

Returns true if two `OracleIntervalYM` values are not equal; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalYM` that has a value compares greater than an `OracleIntervalYM` that has a null value.
- Two `OracleIntervalYMs` that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

operator <

This static operator determines if the first of two `OracleIntervalYM` values is less than the second.

Declaration

```
// C#  
public static bool operator < (OracleIntervalYM val1, OracleIntervalYM val2);
```

Parameters

- `val1`
The first `OracleIntervalYM`.
- `val2`
The second `OracleIntervalYM`.

Return Value

Returns `true` if the first of two `OracleIntervalYM` values is less than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalYM` that has a value compares greater than an `OracleIntervalYM` that has a null value.
- Two `OracleIntervalYMs` that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

operator <=

This static operator determines if the first of two `OracleIntervalYM` values is less than or equal to the second.

Declaration

```
// C#  
public static bool operator <= (OracleIntervalYM val1, OracleIntervalYM val2);
```

Parameters

- *val1*
The first OracleIntervalYM.
- *val2*
The second OracleIntervalYM.

Return Value

Returns true if the first of two OracleIntervalYM values is less than or equal to the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleIntervalYM that has a value compares greater than an OracleIntervalYM that has a null value.
- Two OracleIntervalYMs that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

operator -

This static operator subtracts one OracleIntervalYM structure from another.

Declaration

```
// C#  
public static OracleIntervalYM operator - (OracleIntervalYM val1, OracleIntervalYM val2);
```

Parameters

- *val1*
The first OracleIntervalYM.
- *val2*
The second OracleIntervalYM.

Return Value

An OracleIntervalYM structure.

Remarks

If either argument has a null value, the returned OracleIntervalYM structure has a null value.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

operator -

This static operator negates an OracleIntervalYM structure.

Declaration

```
// C#  
public static OracleIntervalYM operator - (OracleIntervalYM val);
```

Parameters

- *val*
An OracleIntervalYM.

Return Value

An OracleIntervalYM structure.

Remarks

If the supplied OracleIntervalYM structure has a null value, the returned OracleIntervalYM structure has a null value.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

operator *

This static operator multiplies an OracleIntervalYM value by a number.

Declaration

```
// C#  
public static OracleIntervalYM operator * (OracleIntervalYM val1, int multiplier);
```


Parameters

- *val1*
The first OracleIntervalYM.
- *multiplier*
A multiplier.

Return Value

An OracleIntervalYM structure.

Remarks

If the supplied OracleIntervalYM structure has a null value, the returned OracleIntervalYM structure has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

operator /

This static operator divides an OracleIntervalYM value by a number.

Declaration

```
// C#  
public static OracleIntervalYM operator / (OracleIntervalYM val1, int divisor);
```

Parameters

- *val1*
The first OracleIntervalYM.
- *divisor*
A divisor.

Return Value

An OracleIntervalYM structure.

Remarks

If the supplied OracleIntervalYM structure has a null value, the returned OracleIntervalYM structure has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

OracleIntervalYM Type Conversions

The OracleIntervalYM conversions are listed in [Table 14-85](#).

Table 14-85 OracleIntervalYM Type Conversions

Operator	Description
explicit operator long	Converts an OracleIntervalYM structure to a number
explicit operator OracleIntervalYM	Converts a string to an OracleIntervalYM structure
implicit operator OracleIntervalYM	Converts the number of months to an OracleIntervalYM structure

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

explicit operator long

This type conversion operator converts an OracleIntervalYM to a number that represents the number of months in the time interval.

Declaration

```
// C#
public static explicit operator long (OracleIntervalYM val);
```

Parameters

- *val*
An OracleIntervalYM structure.

Return Value

A long number in months.

Exceptions

OracleNullValueException - The OracleIntervalYM structure has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

explicit operator OracleIntervalYM

This type conversion operator converts the string *intervalStr* to an OracleIntervalYM structure.

Declaration

```
// C#  
public static explicit operator OracleIntervalYM (string intervalStr);
```

Parameters

- *intervalStr*

A string representation of an Oracle INTERVAL YEAR TO MONTH.

Return Value

An OracleIntervalYM structure.

Exceptions

ArgumentException - The supplied *intervalStr* parameter is not in the correct format or has an invalid value.

ArgumentNullException - The *intervalStr* parameter is null.

Remarks

The returned OracleIntervalDS structure contains the same time interval represented by the supplied *intervalStr*. The value specified in the supplied *intervalStr* must be in Year-Month format.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

implicit operator OracleIntervalYM

This type conversion operator converts the total number of months as time interval to an OracleIntervalYM structure.

Declaration

```
// C#  
public static implicit operator OracleIntervalYM (long months);
```

Parameters

- *months*
The number of months to be converted. Range is $(-999,999,999 * 12) - 11 \leq months \leq (999,999,999 * 12) + 11$.

Return Value

An OracleIntervalYM structure.

Exceptions

ArgumentOutOfRangeException - The *months* parameter is out of the specified range.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

OracleIntervalYM Properties

The OracleIntervalYM properties are listed in [Table 14-86](#).

Table 14-86 OracleIntervalYM Properties

Properties	Description
BinData	Returns an array of bytes that represents the Oracle INTERVAL YEAR TO MONTH in an Oracle internal format
IsNull	Indicates whether or not the current instance has a null value
Months	Gets the months component of an OracleIntervalYM
TotalYears	Returns the total number, in years, that represents the period of time in the current OracleIntervalYM structure
Value	Specifies the total number of months that is stored in the OracleIntervalYM structure
Years	Gets the years component of an OracleIntervalYM

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

BinData

This property returns an array of bytes that represents the Oracle INTERVAL YEAR TO MONTH in Oracle internal format.

Declaration

```
// C#  
public byte[] BinData {get;}
```

Property Value

A byte array that represents an Oracle INTERVAL YEAR TO MONTH in Oracle internal format.

Exceptions

OracleNullValueException - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

IsNull

This property indicates whether or not the value has a null value.

Declaration

```
// C#  
public bool IsNull {get;}
```

Property Value

Returns true if value has a null value; otherwise, returns false.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

Months

This property gets the months component of an OracleIntervalYM.

Declaration

```
// C#  
public int Months {get;}
```

Property Value

An int representing the months component.

Exceptions

OracleNullValueException - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

TotalYears

This property returns the total number, in years, that represents the period of time in the current OracleIntervalYM structure.

Declaration

```
// C#  
public double TotalYears {get;}
```

Property Value

A double representing the total number of years.

Exceptions

OracleNullValueException - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

Value

This property gets the total number of months that is stored in the OracleIntervalYM structure.

Declaration

```
// C#  
public long Value {get;}
```

Property Value

The total number of months representing the time interval.

Exceptions

OracleNullValueException - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

Years

This property gets the years component of an OracleIntervalYM.

Declaration

```
// C#  
public int Years {get;}
```

Property Value

An int representing the years component.

Exceptions

OracleNullValueException - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

OracleIntervalYM Methods

The OracleIntervalYM methods are listed in [Table 14-87](#).

Table 14-87 OracleIntervalYM Methods

Methods	Description
CompareTo	Compares the current OracleIntervalYM instance to the supplied object, and returns an integer that represents their relative values
Equals	Determines whether or not the specified object has the same time interval as the current instance (Overloaded)
GetHashCode	Returns a hash code for the OracleIntervalYM instance
GetType	Inherited from System.Object
ToString	Converts the current OracleIntervalYM structure to a string

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

CompareTo

This method compares the current OracleIntervalYM instance to the supplied object, and returns an integer that represents their relative values.

Declaration

```
// C#  
public int CompareTo(object obj);
```

Parameters

- *obj*
The supplied object.

Return Value

The method returns a number:

Less than zero: if the current OracleIntervalYM represents a shorter time interval than *obj*.

Zero: if the current OracleIntervalYM and *obj* represent the same time interval.

Greater than zero: if the current OracleIntervalYM represents a longer time interval than *obj*.

Implements

IComparable

Exceptions

ArgumentException - The *obj* parameter is not of type OracleIntervalYM.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between OracleIntervalYMs. For example, comparing an OracleIntervalYM instance with an OracleBinary instance is not allowed. When an OracleIntervalYM is compared with a different type, an ArgumentException is thrown.
- Any OracleIntervalYM that has a value compares greater than an OracleIntervalYM that has a null value.
- Two OracleIntervalYMs that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

Equals

Overrides Object

This method determines whether or not the specified object has the same time interval as the current instance.

Declaration

```
// C#  
public override bool Equals(object obj);
```

Parameters

- *obj*
The supplied object.

Return Value

Returns true if the specified object instance is of type OracleIntervalYM and has the same time interval; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleIntervalYM that has a value compares greater than an OracleIntervalYM that has a null value.
- Two OracleIntervalYMs that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

GetHashCode

Overrides Object

This method returns a hash code for the OracleIntervalYM instance.

Declaration

```
// C#  
public override int GetHashCode();
```

Return Value

An int representing a hash code.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

ToString

Overrides Object

This method converts the current OracleIntervalYM structure to a string.

Declaration

```
// C#  
public override string ToString();
```

Return Value

A string that represents the current OracleIntervalYM structure.

Remarks

If the current instance has a null value, the returned string contain "null".

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleIntervalYM Structure](#)
- [OracleIntervalYM Members](#)

OracleString Structure

The OracleString structure represents a variable-length stream of characters to be stored in or retrieved from a database.

Class Inheritance

System.Object
 System.ValueType
 Oracle.DataAccess.Types.OracleString

Declaration

```
// C#  
public struct OracleString : IComparable, INullable, IXmlSerializable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Types;
```

```
class OracleStringSample
{
    static void Main()
    {
        // Initialize OracleString structs
        OracleString string1 = new OracleString("AAA");

        // Display the string "AAA"
        Console.WriteLine("{0} has length of {1}", string1, string1.Length);

        // Contatenate characters to string1 until the length is 5
        while (string1.Length < 5)
            string1 = OracleString.Concat(string1, "a");

        // Display the string of "AAaaa"
        Console.WriteLine("{0} has length of {1}", string1, string1.Length);
    }
}
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Members](#)
- [OracleString Constructors](#)
- [OracleString Static Fields](#)
- [OracleString Static Methods](#)
- [OracleString Static Operators](#)
- [OracleString Type Conversions](#)
- [OracleString Properties](#)
- [OracleString Methods](#)

OracleString Members

OracleString members are listed in the following tables:

OracleString Constructors

OracleString constructors are listed in [Table 14-88](#)

Table 14-88 OracleString Constructors

Constructor	Description
OracleString Constructors	Instantiates a new instance of OracleString structure (Overloaded)

OracleString Static Fields

The OracleString static fields are listed in [Table 14-89](#).

Table 14-89 OracleString Static Fields

Field	Description
Null	Represents a null value that can be assigned to an instance of the OracleString structure

OracleString Static Methods

The OracleString static methods are listed in [Table 14-90](#).

Table 14-90 OracleString Static Methods

Methods	Description
Concat	Concatenates two OracleString instances and returns a new OracleString instance that represents the result
Equals	Determines if two OracleString values are equal (Overloaded)
GreaterThan	Determines whether or not the first of two OracleString values is greater than the second
GreaterThanOrEqual	Determines whether or not the first of two OracleString values is greater than or equal to the second
LessThan	Determines whether or not the first of two OracleString values is less than the second
LessThanOrEqual	Determines whether or not the first of two OracleString values is less than or equal to the second
NotEquals	Determines whether two OracleString values are not equal

OracleString Static Operators

The OracleString static operators are listed in [Table 14-91](#).

Table 14-91 OracleString Static Operators

Operator	Description
operator +	Concatenates two OracleString values
operator ==	Determines if two OracleString values are equal
operator >	Determines if the first of two OracleString values is greater than the second
operator >=	Determines if the first of two OracleString values is greater than or equal to the second
operator !=	Determines if the two OracleString values are not equal
operator <	Determines if the first of two OracleString values is less than the second

Table 14-91 (Cont.) OracleString Static Operators

Operator	Description
operator <=	Determines if two OracleString values are not equal

OracleString Type Conversions

The OracleString type conversions are listed in [Table 14-92](#).

Table 14-92 OracleString Type Conversions

Operator	Description
explicit operator string	Converts the supplied OracleString to a string instance
implicit operator OracleString	Converts the supplied string to an OracleString instance

OracleString Properties

The OracleString properties are listed in [Table 14-93](#).

Table 14-93 OracleString Properties

Properties	Description
IsCaseIgnored	Indicates whether or not case should be ignored when performing string comparison
IsNull	Indicates whether or not the current instance has a null value
Item	Obtains the particular character in an OracleString using an index.
Length	Returns the length of the OracleString
Value	Returns the string data that is stored in the OracleString structure.

OracleString Methods

The OracleString methods are listed in [Table 14-94](#).

Table 14-94 OracleString Methods

Methods	Description
Clone	Returns a copy of the current OracleString instance
CompareTo	Compares the current OracleString instance to the supplied object, and returns an integer that represents their relative values
Equals	Determines whether or not an object has the same string value as the current OracleString structure (Overloaded)

Table 14-94 (Cont.) OracleString Methods

Methods	Description
GetHashCode	Returns a hash code for the OracleString instance
GetNonUnicodeBytes	Returns an array of bytes, containing the contents of the OracleString, in the client character set format
GetType	Inherited from System.Object
GetUnicodeBytes	Returns an array of bytes, containing the contents of the OracleString, in Unicode format
ToString	Converts the current OracleString instance to a string

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)

OracleString Constructors

The OracleString constructors create new instances of the OracleString structure.

Overload List:

- [OracleString\(string\)](#)
This constructor creates a new instance of the OracleString structure and sets its value using a string.
- [OracleString\(string, bool\)](#)
This constructor creates a new instance of the OracleString structure and sets its value using a string and specifies if case is ignored in comparison.
- [OracleString\(byte \[\], bool\)](#)
This constructor creates a new instance of the OracleString structure and sets its value using a byte array and specifies if the supplied byte array is Unicode encoded.
- [OracleString\(byte \[\], bool, bool\)](#)
This constructor creates a new instance of the OracleString structure and sets its value using a byte array and specifies the following: if the supplied byte array is Unicode encoded and if case is ignored in comparison.
- [OracleString\(byte \[\], int, int, bool\)](#)
This constructor creates a new instance of the OracleString structure and sets its value using a byte array, and specifies the following: the starting index in the byte array, the number of bytes to copy from the byte array, and if the supplied byte array is Unicode encoded.
- [OracleString\(byte \[\], int, int, bool, bool\)](#)

This constructor creates a new instance of the `OracleString` structure and sets its value using a byte array, and specifies the following: the starting index in the byte array, the number of bytes to copy from the byte array, if the supplied byte array is Unicode encoded, and if case is ignored in comparison.

📘 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

OracleString(string)

This constructor creates a new instance of the `OracleString` structure and sets its value using a string.

Declaration

```
// C#  
public OracleString(string data);
```

Parameters

- *data*
A string value.

📘 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

OracleString(string, bool)

This constructor creates a new instance of the `OracleString` structure and sets its value using a string and specifies if case is ignored in comparison.

Declaration

```
// C#  
public OracleString(string data, bool isCaseIgnored);
```

Parameters

- *data*
A string value.
- *isCaseIgnored*

Specifies if case is ignored in comparison. Specifies true if case is to be ignored; otherwise, specifies false.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

OracleString(byte [], bool)

This constructor creates a new instance of the `OracleString` structure and sets its value using a byte array and specifies if the supplied byte array is Unicode encoded.

Declaration

```
// C#  
public OracleString(byte[] data, bool fUnicode);
```

Parameters

- *data*
Byte array data for the new `OracleString`.
- *fUnicode*
Specifies if the supplied data is Unicode encoded. Specifies true if Unicode encoded; otherwise, false.

Exceptions

`ArgumentNullException` - The *data* parameter is null.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

OracleString(byte [], bool, bool)

This constructor creates a new instance of the `OracleString` structure and sets its value using a byte array and specifies the following: if the supplied byte array is Unicode encoded and if case is ignored in comparison.

Declaration

```
// C#  
public OracleString(byte[] data, bool fUnicode, bool isCaseIgnored);
```

Parameters

- *data*
Byte array data for the new OracleString.
- *fUnicode*
Specifies if the supplied data is Unicode encoded. Specifies true if Unicode encoded; otherwise, false.
- *isCaseIgnored*
Specifies if case is ignored in comparison. Specifies true if case is to be ignored; otherwise, specifies false.

Exceptions

ArgumentNullException - The *data* parameter is null.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

OracleString(byte [], int, int, bool)

This constructor creates a new instance of the OracleString structure and sets its value using a byte array, and specifies the following: the starting index in the byte array, the number of bytes to copy from the byte array, and if the supplied byte array is Unicode encoded.

Declaration

```
// C#  
public OracleString(byte[] data, int index, int count, bool fUnicode);
```

Parameters

- *data*
Byte array data for the new OracleString.
- *index*
The starting index to copy from *data*.
- *count*
The number of bytes to copy.
- *fUnicode*
Specifies if the supplied data is Unicode encoded. Specifies true if Unicode encoded; otherwise, false.

Exceptions

ArgumentNullException - The *data* parameter is null.

ArgumentOutOfRangeException - The *count* parameter is less than zero.

IndexOutOfRangeException - The *index* parameter is greater than or equal to the length of *data* or less than zero.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

OracleString(byte [], int, int, bool, bool)

This constructor creates a new instance of the `OracleString` structure and sets its value using a byte array, and specifies the following: the starting index in the byte array, the number of bytes to copy from the byte array, if the supplied byte array is Unicode encoded, and if case is ignored in comparison.

Declaration

```
// C#  
public OracleString(byte[] data, int index, int count, bool fUnicode,  
    bool isCaseIgnored);
```

Parameters

- *data*
Byte array data for the new `OracleString`.
- *index*
The starting index to copy from *data*.
- *count*
The number of bytes to copy.
- *fUnicode*
Specifies if the supplied data is Unicode encoded. Specifies true if Unicode encoded; otherwise, false.
- *isCaseIgnored*
Specifies if case is ignored in comparison. Specifies true if case is to be ignored; otherwise, specifies false.

Exceptions

ArgumentNullException - The *data* parameter is null.

ArgumentOutOfRangeException - The *count* parameter is less than zero.

IndexOutOfRangeException - The *index* parameter is greater than or equal to the length of *data* or less than zero.

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

OracleString Static Fields

The OracleString static fields are listed in [Table 14-95](#).

Table 14-95 OracleString Static Fields

Field	Description
Null	Represents a null value that can be assigned to an instance of the OracleString structure

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

Null

This static field represents a null value that can be assigned to an instance of the OracleString structure.

Declaration

```
// C#  
public static readonly OracleString Null;
```

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

OracleString Static Methods

The OracleString static methods are listed in [Table 14-96](#).

Table 14-96 OracleString Static Methods

Methods	Description
Concat	Concatenates two OracleString instances and returns a new OracleString instance that represents the result
Equals	Determines if two OracleString values are equal (Overloaded)
GreaterThan	Determines whether or not the first of two OracleString values is greater than the second
GreaterThanOrEqual	Determines whether or not the first of two OracleString values is greater than or equal to the second
LessThan	Determines whether or not the first of two OracleString values is less than the second
LessThanOrEqual	Determines whether or not the first of two OracleString values is less than or equal to the second
NotEquals	Determines whether two OracleString values are not equal

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

Concat

This static method concatenates two OracleString instances and returns a new OracleString instance that represents the result.

Declaration

```
// C#  
public static OracleString Concat(OracleString str1, OracleString str2);
```

Parameters

- *str1*
The first OracleString.
- *str2*
The second OracleString.

Return Value

An OracleString.

Remarks

If either argument has a null value, the returned `OracleString` structure has a null value.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleString Structure](#)
- [OracleString Members](#)

Equals

Overloads Object

This static method determines whether or not the two `OracleStrings` being compared are equal.

Declaration

```
// C#  
public static bool Equals(OracleString str1, OracleString str2);
```

Parameters

- *str1*
The first `OracleString`.
- *str2*
The second `OracleString`.

Return Value

Returns true if the two `OracleStrings` being compared are equal; returns false otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleString` that has a value is greater than an `OracleString` that has a null value.
- Two `OracleStrings` that contain a null value are equal.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleString Structure](#)
- [OracleString Members](#)

GreaterThan

This static method determines whether or not the first of two `OracleString` values is greater than the second.

Declaration

```
// C#  
public static bool GreaterThan(OracleString str1, OracleString str2);
```

Parameters

- *str1*
The first `OracleString`.
- *str2*
The second `OracleString`.

Return Value

Returns true if the first of two `OracleStrings` is greater than the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleString` that has a value is greater than an `OracleString` that has a null value.
- Two `OracleStrings` that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

GreaterThanOrEqual

This static method determines whether or not the first of two `OracleString` values is greater than or equal to the second.

Declaration

```
// C#  
public static bool GreaterThanOrEqual(OracleString str1,  
    OracleString str2);
```

Parameters

- *str1*
The first `OracleString`.

- *str2*
The second `OracleString`.

Return Value

Returns `true` if the first of two `OracleStrings` is greater than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleString` that has a value is greater than an `OracleString` that has a null value.
- Two `OracleStrings` that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

LessThan

This static method determines whether or not the first of two `OracleString` values is less than the second.

Declaration

```
// C#  
public static bool LessThan(OracleString str1, OracleString str2);
```

Parameters

- *str1*
The first `OracleString`.
- *str2*
The second `OracleString`.

Return Value

Returns `true` if the first is less than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleString` that has a value is greater than an `OracleString` that has a null value.
- Two `OracleStrings` that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

LessThanOrEqual

This static method determines whether or not the first of two `OracleString` values is less than or equal to the second.

Declaration

```
// C#  
public static bool LessThanOrEqual(OracleString str1, OracleString str2);
```

Parameters

- *str1*
The first `OracleString`.
- *str2*
The second `OracleString`.

Return Value

Returns `true` if the first is less than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleString` that has a value is greater than an `OracleString` that has a null value.
- Two `OracleStrings` that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

NotEquals

This static method determines whether two `OracleString` values are not equal.

Declaration

```
// C#  
public static bool NotEquals(OracleString str1, OracleString str2);
```

Parameters

- *str1*
The first OracleString.
- *str2*
The second OracleString.

Return Value

Returns true if the two OracleString instances are not equal; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleString that has a value is greater than an OracleString that has a null value.
- Two OracleStrings that contain a null value are equal.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleString Structure](#)
- [OracleString Members](#)

OracleString Static Operators

The OracleString static operators are listed in [Table 14-97](#).

Table 14-97 OracleString Static Operators

Operator	Description
operator +	Concatenates two OracleString values
operator ==	Determines if two OracleString values are equal
operator >	Determines if the first of two OracleString values is greater than the second
operator >=	Determines if the first of two OracleString values is greater than or equal to the second
operator !=	Determines if the two OracleString values are not equal
operator <	Determines if the first of two OracleString values is less than the second
operator <=	Determines if two OracleString values are not equal

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

operator +

This static operator concatenates two `OracleString` values.

Declaration

```
// C#
public static OracleString operator + (OracleString value1, OracleString value2);
```

Parameters

- *value1*
The first `OracleString`.
- *value2*
The second `OracleString`.

Return Value

An `OracleString`.

Remarks

If either argument has a null value, the returned `OracleString` structure has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

operator ==

This static operator determines if two `OracleString` values are equal.

Declaration

```
// C#
public static bool operator == (OracleString value1, OracleString value2);
```

Parameters

- *value1*

The first `OracleString`.

- `value2`

The second `OracleString`.

Return Value

Returns true if two `OracleString` values are equal; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleString` that has a value is greater than an `OracleString` that has a null value.
- Two `OracleStrings` that contain a null value are equal.

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleString Structure](#)
- [OracleString Members](#)

operator >

This static operator determines if the first of two `OracleString` values is greater than the second.

Declaration

```
// C#  
public static bool operator > (OracleString value1, OracleString value2);
```

Parameters

- `value1`
The first `OracleString`.
- `value2`
The second `OracleString`.

Return Value

Returns true if the first of two `OracleString` values is greater than the second; otherwise returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleString` that has a value is greater than an `OracleString` that has a null value.
- Two `OracleStrings` that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

operator >=

This static operator determines if the first of two `OracleString` values is greater than or equal to the second.

Declaration

```
// C#  
public static bool operator >= (OracleString value1, OracleString value2);
```

Parameters

- *value1*
The first `OracleString`.
- *value2*
The second `OracleString`.

Return Value

Returns `true` if the first of two `OracleString` values is greater than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleString` that has a value is greater than an `OracleString` that has a null value.
- Two `OracleStrings` that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

operator !=

This static operator determines if two `OracleString` values are not equal.

Declaration

```
// C#  
public static bool operator != (OracleString value1, OracleString value2);
```

Parameters

- *value1*
The first OracleString.
- *value2*
The second OracleString.

Return Value

Returns true if two OracleString values are not equal; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleString that has a value is greater than an OracleString that has a null value.
- Two OracleStrings that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

operator <

This static operator determines if the first of two OracleStrings is less than the second.

Declaration

```
// C#  
public static bool operator < (OracleString value1, OracleString value2);
```

Parameters

- *value1*
The first OracleString.
- *value2*
The second OracleString.

Return Value

Returns true if the first of two OracleStrings is less than the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleString` that has a value is greater than an `OracleString` has a null value.
- Two `OracleStrings` that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

operator <=

This static operator determines if the first of two `OracleString` values is less than or equal to the second.

Declaration

```
// C#  
public static bool operator <= (OracleString value1, OracleString value2);
```

Parameters

- *value1*
The first `OracleString`.
- *value2*
The second `OracleString`.

Return Value

Returns `true` if the first of two `OracleString` values is less than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleString` that has a value is greater than an `OracleString` that has a null value.
- Two `OracleStrings` that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

OracleString Type Conversions

The OracleString type conversions are listed in [Table 14-98](#).

Table 14-98 OracleString Type Conversions

Operator	Description
explicit operator string	Converts the supplied OracleString to a string instance
implicit operator OracleString	Converts the supplied string to an OracleString instance

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

explicit operator string

This type conversion operator converts the supplied OracleString to a string.

Declaration

```
//C#  
public static explicit operator string (OracleString value1);
```

Parameters

- *value1*
The supplied OracleString.

Return Value

string

Exceptions

OracleNullValueException - The OracleString structure has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

implicit operator OracleString

This type conversion operator converts the supplied string to an `OracleString`.

Declaration

```
// C#
public static implicit operator OracleString (string value1);
```

Parameters

- `value1`
The supplied string.

Return Value

An `OracleString`.

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

OracleString Properties

The `OracleString` properties are listed in [Table 14-99](#).

Table 14-99 OracleString Properties

Properties	Description
IsCaseIgnored	Indicates whether or not case should be ignored when performing string comparison
IsNull	Indicates whether or not the current instance has a null value
Item	Obtains the particular character in an <code>OracleString</code> using an index.
Length	Returns the length of the <code>OracleString</code>
Value	Returns the string data that is stored in the <code>OracleString</code> structure.

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

IsCaseIgnored

This property indicates whether or not case should be ignored when performing string comparison.

Declaration

```
//C#  
public bool IsCaseIgnored {get;set;}
```

Property Value

Returns true if string comparison must ignore case; otherwise false.

Remarks

Default value is true.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Types;  
  
class IsCaseIgnoredSample  
{  
    static void Main()  
    {  
        OracleString string1 = new OracleString("aAaAa");  
        OracleString string2 = new OracleString("AaAaA");  
  
        // Ignore case for comparisons  
        string1.IsCaseIgnored = true;  
        string2.IsCaseIgnored = true;  
  
        // Same; Prints 0  
        Console.WriteLine(string1.CompareTo(string2));  
  
        // Make comparisons case sensitive  
        // Note that IsCaseIgnored must be set to false for both  
        //   OracleStrings; otherwise an exception is thrown  
        string1.IsCaseIgnored = false;  
        string2.IsCaseIgnored = false;  
  
        // Different; Prints nonzero value  
        Console.WriteLine(string1.CompareTo(string2));  
    }  
}
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

IsNull

This property indicates whether or not the current instance contains a null value.

Declaration

```
// C#  
public bool IsNull {get;}
```

Property Value

Returns true if the current instance contains has a null value; otherwise, returns false.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

Item

This property obtains the particular character in an OracleString using an index.

Declaration

```
// C#  
public char Item {get;}
```

Property Value

A char value.

Exceptions

OracleNullValueException - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

Length

This property returns the length of the OracleString.

Declaration

```
// C#  
public int Length {get;}
```

Property Value

A int value.

Exceptions

OracleNullValueException - The current instance has a null value.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleString Structure](#)
- [OracleString Members](#)

Value

This property returns the string data that is stored in the OracleString.

Declaration

```
// C#  
public string Value {get;}
```

Property Value

The stored string value

Exceptions

OracleNullValueException - The current instance has a null value.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleString Structure](#)
- [OracleString Members](#)

OracleString Methods

The OracleString methods are listed in [Table 14-100](#).

Table 14-100 OracleString Methods

Methods	Description
Clone	Returns a copy of the current OracleString instance
CompareTo	Compares the current OracleString instance to the supplied object, and returns an integer that represents their relative values
Equals	Determines whether or not an object has the same string value as the current OracleString structure (Overloaded)
GetHashCode	Returns a hash code for the OracleString instance
GetNonUnicodeBytes	Returns an array of bytes, containing the contents of the OracleString, in the client character set format
GetType	Inherited from System.Object
GetUnicodeBytes	Returns an array of bytes, containing the contents of the OracleString, in Unicode format
ToString	Converts the current OracleString instance to a string

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

Clone

This method creates a copy of an OracleString instance.

Declaration

```
// C#  
public OracleString Clone();
```

Return Value

An OracleString structure.

Remarks

The cloned object has the same property values as that of the object being cloned.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Types;  
  
class CloneSample  
{  
    static void Main()  
    {  

```

```
OracleString str1 = new OracleString("aAaAa");
OracleString str2 = str1.Clone();

// The OracleStrings are same; Prints 0
Console.WriteLine(str1.CompareTo(str2));
}
}
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

CompareTo

This method compares the current `OracleString` instance to the supplied object, and returns an integer that represents their relative values.

Declaration

```
// C#
public int CompareTo(object obj);
```

Parameters

- *obj*
The object being compared to the current instance.

Return Value

The method returns a number that is:

- Less than zero: if the current `OracleString` value is less than *obj*.
- Zero: if the current `OracleString` value is equal to *obj*.
- Greater than zero: if the current `OracleString` value is greater than *obj*.

Implements

`IComparable`

Exceptions

`ArgumentException` - The *obj* parameter is not of type `OracleString`.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between `OracleStrings`. For example, comparing an `OracleString` instance with an `OracleBinary` instance is not allowed. When an `OracleString` is compared with a different type, an `ArgumentException` is thrown.
- Any `OracleString` that has a value is greater than an `OracleString` that has a null value.

- Two OracleStrings that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

Equals

This method determines whether or not supplied object is an instance of OracleString and has the same values as the current OracleString instance.

Declaration

```
// C#  
public override bool Equals(object obj);
```

Parameters

- *obj*
An object being compared.

Return Value

Returns true if the supplied object is an instance of OracleString and has the same values as the current OracleString instance; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleString that has a value is greater than an OracleString that has a null value.
- Two OracleStrings that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

GetHashCode

Overrides Object

This method returns a hash code for the OracleString instance.

Declaration

```
// C#  
public override int GetHashCode();
```

Return Value

A number that represents the hash code.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

GetNonUnicodeBytes

This method returns an array of bytes, containing the contents of the `OracleString`, in the client character set format.

Declaration

```
// C#  
public byte[] GetNonUnicodeBytes();
```

Return Value

A byte array that contains the contents of the `OracleString` in the client character set format.

Remarks

If the current instance has a null value, an `OracleNullValueException` is thrown.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

GetUnicodeBytes

This method returns an array of bytes, containing the contents of the `OracleString` in Unicode format.

Declaration

```
// C#  
public byte[] GetUnicodeBytes();
```


Return Value

A byte array that contains the contents of the `OracleString` in Unicode format.

Remarks

If the current instance has a null value, an `OracleNullValueException` is thrown.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

ToString

Overrides `Object`

This method converts the current `OracleString` instance to a string.

Declaration

```
// C#  
public override string ToString();
```

Return Value

A string.

Remarks

If the current `OracleString` instance has a null value, the string contains "null".

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleString Structure](#)
- [OracleString Members](#)

OracleTimeStamp Structure

The `OracleTimeStamp` structure represents the Oracle `TIMESTAMP` data type to be stored in or retrieved from a database. Each `OracleTimeStamp` stores the following information: year, month, day, hour, minute, second, and nanosecond.

Class Inheritance

`System.Object`

System.ValueType
Oracle.DataAccess.Types.OracleTimeStamp

Declaration

```
// C#public struct OracleTimeStamp : IComparable, INullable, IXmlSerializable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Types;  
  
class OracleTimeStampSample  
{  
    static void Main()  
    {  
        OracleTimeStamp tsCurrent1 = OracleTimeStamp.GetSysDate();  
        OracleTimeStamp tsCurrent2 = DateTime.Now;  
  
        // Calculate the difference between tsCurrent1 and tsCurrent2  
        OracleIntervalDS idsDiff = tsCurrent2.GetDaysBetween(tsCurrent1);  
  
        // Calculate the difference using AddNanoseconds()  
        int nanoDiff = 0;  
        while (tsCurrent2 > tsCurrent1)  
        {  
            nanoDiff += 10;  
            tsCurrent1 = tsCurrent1.AddNanoseconds(10);  
        }  
        Console.WriteLine("idsDiff.Nanoseconds = " + idsDiff.Nanoseconds);  
        Console.WriteLine("nanoDiff = " + nanoDiff);  
    }  
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Members](#)
- [OracleTimeStamp Constructors](#)
- [OracleTimeStamp Static Fields](#)
- [OracleTimeStamp Static Methods](#)
- [OracleTimeStamp Static Operators](#)
- [OracleTimeStamp Static Type Conversions](#)
- [OracleTimeStamp Properties](#)
- [OracleTimeStamp Methods](#)

OracleTimeStamp Members

OracleTimeStamp members are listed in the following tables:

OracleTimeStamp Constructors

OracleTimeStamp constructors are listed in [Table 14-101](#)

Table 14-101 OracleTimeStamp Constructors

Constructor	Description
OracleTimeStamp Constructors	Instantiates a new instance of OracleTimeStamp structure (Overloaded)

OracleTimeStamp Static Fields

The OracleTimeStamp static fields are listed in [Table 14-102](#).

Table 14-102 OracleTimeStamp Static Fields

Field	Description
MaxValue	Represents the maximum valid date for an OracleTimeStamp structure, which is December 31, 9999 23:59:59.999999999
MinValue	Represents the minimum valid date for an OracleTimeStamp structure, which is January 1, -4712 0:0:0
Null	Represents a null value that can be assigned to an instance of the OracleTimeStamp structure

OracleTimeStamp Static Methods

The OracleTimeStamp static methods are listed in [Table 14-103](#).

Table 14-103 OracleTimeStamp Static Methods

Methods	Description
Equals	Determines if two OracleTimeStamp values are equal (Overloaded)
GreaterThan	Determines if the first of two OracleTimeStamp values is greater than the second
GreaterThanOrEqual	Determines if the first of two OracleTimeStamp values is greater than or equal to the second
LessThan	Determines if the first of two OracleTimeStamp values is less than the second
LessThanOrEqual	Determines if the first of two OracleTimeStamp values is less than or equal to the second
NotEquals	Determines if two OracleTimeStamp values are not equal
GetSysDate	Gets an OracleTimeStamp structure that represents the current date and time
Parse	Gets an OracleTimeStamp structure and sets its value using the supplied string
SetPrecision	Returns a new instance of an OracleTimeStamp with the specified fractional second precision

OracleTimeStamp Static Operators

The OracleTimeStamp static operators are listed in [Table 14-104](#).

Table 14-104 OracleTimeStamp Static Operators

Operator	Description
operator +	Adds the supplied instance value to the supplied OracleTimeStamp and returns a new OracleTimeStamp structure (Overloaded)
operator ==	Determines if two OracleTimeStamp values are equal
operator >	Determines if the first of two OracleTimeStamp values is greater than the second
operator >=	Determines if the first of two OracleTimeStamp values is greater than or equal to the second
operator !=	Determines if the two OracleTimeStamp values are not equal
operator <	Determines if the first of two OracleTimeStamp values is less than the second
operator <=	Determines if the first of two OracleTimeStamp values is less than or equal to the second
operator -	Subtracts the supplied instance value from the supplied OracleTimeStamp and returns a new OracleTimeStamp structure (Overloaded)

OracleTimeStamp Static Type Conversions

The OracleTimeStamp static type conversions are listed in [Table 14-105](#).

Table 14-105 OracleTimeStamp Static Type Conversions

Operator	Description
explicit operator OracleTimeStamp	Converts an instance value to an OracleTimeStamp structure (Overloaded)
implicit operator OracleTimeStamp	Converts an instance value to an OracleTimeStamp structure (Overloaded)
explicit operator DateTime	Converts an OracleTimeStamp value to a DateTime structure

OracleTimeStamp Properties

The OracleTimeStamp properties are listed in [Table 14-106](#).

Table 14-106 OracleTimeStamp Properties

Properties	Description
BinData	Returns an array of bytes that represents an Oracle TIMESTAMP in Oracle internal format
Day	Specifies the day component of an OracleTimeStamp
IsNull	Indicates whether or not the OracleTimeStamp instance has a null value
Hour	Specifies the hour component of an OracleTimeStamp
Millisecond	Specifies the millisecond component of an OracleTimeStamp
Minute	Specifies the minute component of an OracleTimeStamp
Month	Specifies the month component of an OracleTimeStamp
Nanosecond	Specifies the nanosecond component of an OracleTimeStamp
Second	Specifies the second component of an OracleTimeStamp
Value	Specifies the date and time that is stored in the OracleTimeStamp structure
Year	Specifies the year component of an OracleTimeStamp

OracleTimeStamp Methods

The OracleTimeStamp methods are listed in [Table 14-107](#).

Table 14-107 OracleTimeStamp Methods

Methods	Description
AddDays	Adds the supplied number of days to the current instance
AddHours	Adds the supplied number of hours to the current instance
AddMilliseconds	Adds the supplied number of milliseconds to the current instance
AddMinutes	Adds the supplied number of minutes to the current instance
AddMonths	Adds the supplied number of months to the current instance
AddNanoseconds	Adds the supplied number of nanoseconds to the current instance
AddSeconds	Adds the supplied number of seconds to the current instance
AddYears	Adds the supplied number of years to the current instance
CompareTo	Compares the current OracleTimeStamp instance to an object, and returns an integer that represents their relative values
Equals	Determines whether or not an object has the same date and time as the current OracleTimeStamp instance (Overloaded)
GetHashCode	Returns a hash code for the OracleTimeStamp instance
GetDaysBetween	Subtracts an OracleTimeStamp value from the current instance and returns an OracleIntervalDS that represents the time difference between the supplied OracleTimeStamp and the current instance
GetYearsBetween	Subtracts value1 from the current instance and returns an OracleIntervalYM that represents the difference between value1 and the current instance using OracleIntervalYM
GetType	Inherited from System.Object
ToOracleDate	Converts the current OracleTimeStamp structure to an OracleDate structure
ToOracleTimeStampLTZ	Converts the current OracleTimeStamp structure to an OracleTimeStampLTZ structure
ToOracleTimeStampTZ	Converts the current OracleTimeStamp structure to an OracleTimeStampTZ structure
ToString	Converts the current OracleTimeStamp structure to a string

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)

OracleTimeStamp Constructors

The OracleTimeStamp constructors create new instances of the OracleTimeStamp structure.

Overload List:

- [OracleTimeStamp\(DateTime\)](#)
This constructor creates a new instance of the OracleTimeStamp structure and sets its value for date and time using the supplied DateTime value.
- [OracleTimeStamp\(string\)](#)
This constructor creates a new instance of the OracleTimeStamp structure and sets its value using the supplied string.
- [OracleTimeStamp\(int, int, int\)](#)
This constructor creates a new instance of the OracleTimeStamp structure and sets its value for date using year, month, and day.
- [OracleTimeStamp\(int, int, int, int, int, int\)](#)
This constructor creates a new instance of the OracleTimeStamp structure and sets its value for date and time using year, month, day, hour, minute, and second.
- [OracleTimeStamp\(int, int, int, int, int, int, double\)](#)
This constructor creates a new instance of the OracleTimeStamp structure and sets its value for date and time using year, month, day, hour, minute, second, and millisecond.
- [OracleTimeStamp\(int, int, int, int, int, int, int\)](#)
This constructor creates a new instance of the OracleTimeStamp structure and sets its value for date and time using year, month, day, hour, minute, second, and nanosecond.
- [OracleTimeStamp\(byte \[\]\)](#)
This constructor creates a new instance of the OracleTimeStamp structure and sets its value to the provided byte array, which is in the internal Oracle TIMESTAMP format.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

OracleTimeStamp(DateTime)

This constructor creates a new instance of the `OracleTimeStamp` structure and sets its value for date and time using the supplied `DateTime` value.

Declaration

```
// C#  
public OracleTimeStamp (DateTime dt);
```

Parameters

- *dt*
The supplied `DateTime` value.

Exceptions

`ArgumentException` - The *dt* parameter cannot be used to construct a valid `OracleTimeStamp`.

❶ See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

OracleTimeStamp(string)

This constructor creates a new instance of the `OracleTimeStamp` structure and sets its value using the supplied string.

Declaration

```
// C#  
public OracleTimeStamp (string tsStr);
```

Parameters

- *tsStr*
A string that represents an Oracle `TIMESTAMP`.

Exceptions

`ArgumentException` - The *tsStr* value is an invalid string representation of an Oracle `TIMESTAMP` or the supplied *tsStr* is not in the timestamp format specified by the `OracleGlobalization.TimestampFormat` property of the thread, which represents the Oracle `NLS_TIMESTAMP_FORMAT` parameter.

`ArgumentNullException` - The *tsStr* value is null.

Remarks

The names and abbreviations used for months and days are in the language specified by the `DateLanguage` and `Calendar` properties of the thread's `OracleGlobalization` object. If any of

the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#

using System;
using Oracle.DataAccess.Types;
using Oracle.DataAccess.Client;

class OracleTimeStampSample
{
    static void Main()
    {
        // Set the nls_timestamp_format for the OracleTimeStamp(string)
        // constructor
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.TimestampFormat = "DD-MON-YYYY HH:MI:SS.FF AM";
        OracleGlobalization.SetThreadInfo(info);

        // construct OracleTimeStamp from a string using the format specified.
        OracleTimeStamp ts = new OracleTimeStamp("11-NOV-1999 11:02:33.444 AM");

        // Set the nls_timestamp_format for the ToString() method
        info.TimestampFormat = "YYYY-MON-DD HH:MI:SS.FF AM";
        OracleGlobalization.SetThreadInfo(info);

        // Prints "1999-NOV-11 11:02:33.444000000 AM"
        Console.WriteLine(ts.ToString());
    }
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)
- *Oracle AI Database SQL Language Reference* for further information on date format elements

OracleTimeStamp(int, int, int)

This constructor creates a new instance of the OracleTimeStamp structure and sets its value for date using year, month, and day.

Declaration

```
// C#
public OracleTimeStamp(int year, int month, int day);
```

Parameters

- *year*
The year provided. Range of year is (-4712 to 9999).
- *month*
The month provided. Range of month is (1 to 12).
- *day*
The day provided. Range of day is (1 to 31).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStamp` (that is, the day is out of range for the month).

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

OracleTimeStamp(int, int, int, int, int, int)

This constructor creates a new instance of the `OracleTimeStamp` structure and sets its value for date and time using year, month, day, hour, minute, and second.

Declaration

```
// C#  
public OracleTimeStamp (int year, int month, int day, int hour,  
    int minute, int second);
```

Parameters

- *year*
The year provided. Range of year is (-4712 to 9999).
- *month*
The month provided. Range of *month* is (1 to 12).
- *day*
The day provided. Range of *day* is (1 to 31).
- *hour*
The hour provided. Range of *hour* is (0 to 23).
- *minute*

The minute provided. Range of *minute* is (0 to 59).

- *second*

The second provided. Range of *second* is (0 to 59).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStamp` (that is, the day is out of range for the month).

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

OracleTimeStamp(int, int, int, int, int, int, double)

This constructor creates a new instance of the `OracleTimeStamp` structure and sets its value for date and time using year, month, day, hour, minute, second, and millisecond.

Declaration

```
// C#
public OracleTimeStamp(int year, int month, int day, int hour,
    int minute, int second, double millisecond);
```

Parameters

- *year*
The year provided. Range of year is (-4712 to 9999).
- *month*
The month provided. Range of month is (1 to 12).
- *day*
The day provided. Range of day is (1 to 31).
- *hour*
The hour provided. Range of hour is (0 to 23).
- *minute*
The minute provided. Range of minute is (0 to 59).
- *second*
The second provided. Range of second is (0 to 59).
- *milliseconds*
The milliseconds provided. Range of millisecond is (0 to 999.999999).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStamp` (that is, the day is out of range for the month).

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

OracleTimeStamp(int, int, int, int, int, int, int)

This constructor creates a new instance of the `OracleTimeStamp` structure and sets its value for date and time using year, month, day, hour, minute, second, and nanosecond.

Declaration

```
// C#  
public OracleTimeStamp (int year, int month, int day, int hour,  
    int minute, int second, int nanosecond);
```

Parameters

- *year*
The year provided. Range of *year* is (-4712 to 9999).
- *month*
The month provided. Range of *month* is (1 to 12).
- *day*
The day provided. Range of *day* is (1 to 31).
- *hour*
The hour provided. Range of *hour* is (0 to 23).
- *minute*
The minute provided. Range of *minute* is (0 to 59).
- *second*
The second provided. Range of *second* is (0 to 59).
- *nanosecond*
The nanosecond provided. Range of *nanosecond* is (0 to 999999999).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStamp` (that is, the day is out of range for the month).

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

OracleTimeStamp(byte [])

This constructor creates a new instance of the `OracleTimeStamp` structure and sets its value to the provided byte array, which is in the internal Oracle `TIMESTAMP` format.

Declaration

```
// C#  
public OracleTimeStamp (byte[] bytes);
```

Parameters

- *bytes*
A byte array that represents an Oracle `TIMESTAMP` in Oracle internal format.

Exceptions

`ArgumentException` - *bytes* is not in an internal Oracle `TIMESTAMP` format or *bytes* is not a valid Oracle `TIMESTAMP`.

`ArgumentNullException` - *bytes* is null.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

OracleTimeStamp Static Fields

The `OracleTimeStamp` static fields are listed in [Table 14-108](#).

Table 14-108 OracleTimeStamp Static Fields

Field	Description
MaxValue	Represents the maximum valid date for an OracleTimeStamp structure, which is December 31, 9999 23:59:59.999999999
MinValue	Represents the minimum valid date for an OracleTimeStamp structure, which is January 1, -4712 0:0:0
Null	Represents a null value that can be assigned to an instance of the OracleTimeStamp structure

📘 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

MaxValue

This static field represents the maximum valid date and time for an OracleTimeStamp structure, which is December 31, 9999 23:59:59.999999999.

Declaration

```
// C#
public static readonly OraTimestamp MaxValue;
```

📘 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

MinValue

This static field represents the minimum valid date and time for an OracleTimeStamp structure, which is January 1, -4712 0:0:0.

Declaration

```
// C#
public static readonly OracleTimeStamp MinValue;
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

Null

This static field represents a null value that can be assigned to an instance of the OracleTimeStamp structure.

Declaration

```
// C#  
public static readonly OracleTimeStamp Null;
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

OracleTimeStamp Static Methods

The OracleTimeStamp static methods are listed in [NOT_SUPPORTED](#).

NOT_SUPPORTED

Methods	Description
Equals	Determines if two OracleTimeStamp values are equal (Overloaded)
GreaterThan	Determines if the first of two OracleTimeStamp values is greater than the second
GreaterThanOrEqual	Determines if the first of two OracleTimeStamp values is greater than or equal to the second
LessThan	Determines if the first of two OracleTimeStamp values is less than the second
LessThanOrEqual	Determines if the first of two OracleTimeStamp values is less than or equal to the second
NotEquals	Determines if two OracleTimeStamp values are not equal
GetSysDate	Gets an OracleTimeStamp structure that represents the current date and time
Parse	Gets an OracleTimeStamp structure and sets its value using the supplied string

NOT_SUPPORTED

Methods	Description
SetPrecision	Returns a new instance of an OracleTimeStamp with the specified fractional second precision

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

Equals

This static method determines if two OracleTimeStamp values are equal.

Declaration

```
// C#  
public static bool Equals(OracleTimeStamp value1, OracleTimeStamp value2);
```

Parameters

- *value1*
The first OracleTimeStamp.
- *value2*
The second OracleTimeStamp.

Return Value

Returns true if two OracleTimeStamp values are equal; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleTimeStamp that has a value is greater than an OracleTimeStamp that has a null value.
- Two OracleTimeStamps that contain a null value are equal.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

GreaterThan

This static method determines if the first of two `OracleTimeStamp` values is greater than the second.

Declaration

```
// C#  
public static bool GreaterThan(OracleTimeStamp value1,  
    OracleTimeStamp value2);
```

Parameters

- *value1*
The first `OracleTimeStamp`.
- *value2*
The second `OracleTimeStamp`.

Return Value

Returns `true` if the first of two `OracleTimeStamp` values is greater than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStamp` that has a value is greater than an `OracleTimeStamp` that has a null value.
- Two `OracleTimeStamps` that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

GreaterThanOrEqual

This static method determines if the first of two `OracleTimeStamp` values is greater than or equal to the second.

Declaration

```
// C#  
public static bool GreaterThanOrEqual(OracleTimeStamp value1,  
    OracleTimeStamp value2);
```

Parameters

- *value1*

The first `OracleTimeStamp`.

- `value2`

The second `OracleTimeStamp`.

Return Value

Returns `true` if the first of two `OracleTimeStamp` values is greater than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStamp` that has a value is greater than an `OracleTimeStamp` that has a null value.
- Two `OracleTimestamps` that contain a null value are equal.

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

LessThan

This static method determines if the first of two `OracleTimeStamp` values is less than the second.

Declaration

```
// C#  
public static bool LessThan(OracleTimeStamp value1,  
    OracleTimeStamp value2);
```

Parameters

- `value1`
The first `OracleTimeStamp`.
- `value2`
The second `OracleTimeStamp`.

Return Value

Returns `true` if the first of two `OracleTimeStamp` values is less than the second. Returns `false` otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStamp` that has a value is greater than an `OracleTimeStamp` that has a null value.

- Two OracleTimeStamp values that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

LessThanOrEqual

This static method determines if the first of two OracleTimeStamp values is less than or equal to the second.

Declaration

```
// C#  
public static bool LessThanOrEqual(OracleTimeStamp value1,  
    OracleTimeStamp value2);
```

Parameters

- *value1*
The first OracleTimeStamp.
- *value2*
The second OracleTimeStamp.

Return Value

Returns true if the first of two OracleTimeStamp values is less than or equal to the second. Returns false otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any OracleTimeStamp that has a value is greater than an OracleTimeStamp that has a null value.
- Two OracleTimeStamp values that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

NotEquals

This static method determines if two OracleTimeStamp values are not equal.

Declaration

```
// C#  
public static bool NotEquals(OracleTimeStamp value1,  
    OracleTimeStamp value2);
```

Parameters

- *value1*
The first OracleTimeStamp.
- *value2*
The second OracleTimeStamp.

Return Value

Returns true if two OracleTimeStamp values are not equal. Returns false otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any OracleTimeStamp that has a value is greater than an OracleTimeStamp that has a null value.
- Two OracleTimestamps that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

GetSysDate

This static method gets an OracleTimeStamp structure that represents the current date and time.

Declaration

```
// C#  
public static OracleTimeStamp GetSysDate();
```

Return Value

An OracleTimeStamp structure that represents the current date and time.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

Parse

This static method gets an `OracleTimeStamp` structure and sets its value using the supplied string.

Declaration

```
// C#  
public static OracleTimeStamp Parse(string datetime);
```

Parameters

- *datetime*
A string that represents an Oracle TIMESTAMP.

Return Value

An `OracleTimeStamp` structure.

Exceptions

`ArgumentException` - The *tsStr* is an invalid string representation of an Oracle TIMESTAMP or the supplied *tsStr* is not in the timestamp format specified by the `OracleGlobalization.TimeStampFormat` property of the thread, which represents the Oracle `NLS_TIMESTAMP_FORMAT` parameter.

`ArgumentNullException` - The *tsStr* value is null.

Remarks

The names and abbreviations used for months and days are in the language specified by the `DateLanguage` and `Calendar` properties of the thread's `OracleGlobalization` object. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Types;  
using Oracle.DataAccess.Client;  
  
class ParseSample  
{  
    static void Main()  
    {  
        // Set the nls_timestamp_format for the Parse() method  
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
```

```
info.TimeStampFormat = "DD-MON-YYYY HH:MI:SS.FF AM";
OracleGlobalization.SetThreadInfo(info);

// construct OracleTimeStamp from a string using the format specified.
OracleTimeStamp ts =
    OracleTimeStamp.Parse("11-NOV-1999 11:02:33.444 AM");

// Set the nls_timestamp_format for the ToString() method
info.TimeStampFormat = "YYYY-MON-DD HH:MI:SS.FF AM";
OracleGlobalization.SetThreadInfo(info);

// Prints "1999-NOV-11 11:02:33.444000000 AM"
Console.WriteLine(ts.ToString());
    }
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)

SetPrecision

This static method returns a new instance of an `OracleTimeStamp` with the specified fractional second precision.

Declaration

```
// C#
public static OracleTimeStamp SetPrecision(OracleTimeStamp value1,
    int fracSecPrecision);
```

Parameters

- *value1*
The provided `OracleTimeStamp` object.
- *fracSecPrecision*
The fractional second precision provided. Range of fractional second precision is (0 to 9).

Return Value

An `OracleTimeStamp` structure with the specified fractional second precision.

Exceptions

`ArgumentOutOfRangeException` - *fracSecPrecision* is out of the specified range.

Remarks

The value specified in the supplied *fracSecPrecision* is used to perform a rounding off operation on the supplied `OracleTimeStamp` value. Depending on this value, 0 or more trailing zeros are displayed in the string returned by `ToString()`.

Example

The `OracleTimeStamp` with a value of "December 31, 9999 23:59:59.99" results in the string "December 31, 9999 23:59:59.99000" when `SetPrecision()` is called with the fractional second precision set to 5.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

OracleTimeStamp Static Operators

The `OracleTimeStamp` static operators are listed in [Table 14-110](#).

Table 14-110 OracleTimeStamp Static Operators

Operator	Description
operator +	Adds the supplied instance value to the supplied <code>OracleTimeStamp</code> and returns a new <code>OracleTimeStamp</code> structure (Overloaded)
operator ==	Determines if two <code>OracleTimeStamp</code> values are equal
operator >	Determines if the first of two <code>OracleTimeStamp</code> values is greater than the second
operator >=	Determines if the first of two <code>OracleTimeStamp</code> values is greater than or equal to the second
operator !=	Determines if the two <code>OracleTimeStamp</code> values are not equal
operator <	Determines if the first of two <code>OracleTimeStamp</code> values is less than the second
operator <=	Determines if the first of two <code>OracleTimeStamp</code> values is less than or equal to the second
operator -	Subtracts the supplied instance value from the supplied <code>OracleTimeStamp</code> and returns a new <code>OracleTimeStamp</code> structure (Overloaded)

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

operator +

operator+ adds the supplied object to the OracleTimeStamp and returns a new OracleTimeStamp structure.

Overload List:

- [operator + \(OracleTimeStamp, OracleIntervalDS\)](#)
This static operator adds the supplied OracleIntervalDS to the OracleTimeStamp and returns a new OracleTimeStamp structure.
- [operator + \(OracleTimeStamp, OracleIntervalYM\)](#)
This static operator adds the supplied OracleIntervalYM to the supplied OracleTimeStamp and returns a new OracleTimeStamp structure.
- [operator + \(OracleTimeStamp, TimeSpan\)](#)
This static operator adds the supplied TimeSpan to the supplied OracleTimeStamp and returns a new OracleTimeStamp structure.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

operator + (OracleTimeStamp, OracleIntervalDS)

This static operator adds the supplied OracleIntervalDS to the OracleTimeStamp and returns a new OracleTimeStamp structure.

Declaration

```
// C#  
public static operator + (OracleTimeStamp value1, OracleIntervalDS value2);
```

Parameters

- *value1*
An OracleTimeStamp.
- *value2*
An OracleIntervalDS.

Return Value

An `OracleTimeStamp`.

Remarks

If either parameter has a null value, the returned `OracleTimeStamp` has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

operator + (OracleTimeStamp, OracleIntervalYM)

This static operator adds the supplied `OracleIntervalYM` to the supplied `OracleTimeStamp` and returns a new `OracleTimeStamp` structure.

Declaration

```
// C#  
public static operator + (OracleTimeStamp value1, OracleIntervalYM value2);
```

Parameters

- *value1*
An `OracleTimeStamp`.
- *value2*
An `OracleIntervalYM`.

Return Value

An `OracleTimeStamp`.

Remarks

If either parameter has a null value, the returned `OracleTimeStamp` has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

operator + (OracleTimeStamp, TimeSpan)

This static operator adds the supplied TimeSpan to the supplied OracleTimeStamp and returns a new OracleTimeStamp structure.

Declaration

```
// C#  
public static operator + (OracleTimeStamp value1, TimeSpan value2);
```

Parameters

- *value1*
An OracleTimeStamp.
- *value2*
A TimeSpan.

Return Value

An OracleTimeStamp.

Remarks

If the OracleTimeStamp instance has a null value, the returned OracleTimeStamp has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

operator ==

This static operator determines if two OracleTimeStamp values are equal.

Declaration

```
// C#  
public static bool operator == (OracleTimeStamp value1,  
    OracleTimeStamp value2);
```

Parameters

- *value1*
The first OracleTimeStamp.
- *value2*
The second OracleTimeStamp.

Return Value

Returns true if they are the same; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleTimeStamp that has a value is greater than an OracleTimeStamp that has a null value.
- Two OracleTimeStamps that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

operator >

This static operator determines if the first of two OracleTimeStamp values is greater than the second.

Declaration

```
// C#
public static bool operator > (OracleTimeStamp value1,
    OracleTimeStamp value2);
```

Parameters

- *value1*
The first OracleTimeStamp.
- *value2*
The second OracleTimeStamp.

Return Value

Returns true if the first OracleTimeStamp value is greater than the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleTimeStamp that has a value is greater than an OracleTimeStamp that has a null value.
- Two OracleTimeStamps that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

operator >=

This static operator determines if the first of two `OracleTimeStamp` values is greater than or equal to the second.

Declaration

```
// C#
public static bool operator >= (OracleTimeStamp value1,
    OracleTimeStamp value2);
```

Parameters

- *value1*
The first `OracleTimeStamp`.
- *value2*
The second `OracleTimeStamp`.

Return Value

Returns `true` if the first `OracleTimeStamp` is greater than or equal to the second; otherwise returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStamp` that has a value is greater than an `OracleTimeStamp` that has a null value.
- Two `OracleTimestamps` that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

operator !=

This static operator determines if two `OracleTimeStamp` values are not equal.

Declaration

```
// C#  
public static bool operator != (OracleTimeStamp value1,  
    OracleTimeStamp value2);
```

Parameters

- *value1*
The first OracleTimeStamp.
- *value2*
The second OracleTimeStamp.

Return Value

Returns true if two OracleTimeStamp values are not equal; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleTimeStamp that has a value is greater than an OracleTimeStamp that has a null value.
- Two OracleTimestamps that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

operator <

This static operator determines if the first of two OracleTimeStamp values is less than the second.

Declaration

```
// C#  
public static bool operator < (OracleTimeStamp value1,  
    OracleTimeStamp value2);
```

Parameters

- *value1*
The first OracleTimeStamp.
- *value2*
The second OracleTimeStamp.

Return Value

Returns true if the first `OracleTimeStamp` is less than the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStamp` that has a value is greater than an `OracleTimeStamp` that has a null value.
- Two `OracleTimeStamps` that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

operator <=

This static operator determines if the first of two `OracleTimeStamp` values is less than or equal to the second.

Declaration

```
// C#  
public static bool operator <= (OracleTimeStamp value1,  
    OracleTimeStamp value2);
```

Parameters

- *value1*
The first `OracleTimeStamp`.
- *value2*
The second `OracleTimeStamp`.

Return Value

Returns true if the first `OracleTimeStamp` is less than or equal to the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStamp` that has a value is greater than an `OracleTimeStamp` that has a null value.
- Two `OracleTimeStamps` that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

operator -

operator - subtracts the supplied value, from the supplied OracleTimeStamp value, and returns a new OracleTimeStamp structure.

Overload List:

- [operator - \(OracleTimeStamp, OracleIntervalDS\)](#)
This static operator subtracts the supplied OracleIntervalDS value, from the supplied OracleTimeStamp value, and return a new OracleTimeStamp structure.
- [operator - \(OracleTimeStamp, OracleIntervalYM\)](#)
This static operator subtracts the supplied OracleIntervalYM value, from the supplied OracleTimeStamp value, and returns a new OracleTimeStamp structure.
- [operator - \(OracleTimeStamp, TimeSpan\)](#)
This static operator subtracts the supplied TimeSpan value, from the supplied OracleTimeStamp value, and returns a new OracleTimeStamp structure.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

operator - (OracleTimeStamp, OracleIntervalDS)

This static operator subtracts the supplied OracleIntervalDS value, from the supplied OracleTimeStamp value, and return a new OracleTimeStamp structure.

Declaration

```
// C#  
public static operator - (OracleTimeStamp value1, OracleIntervalDS value2);
```

Parameters

- *value1*
An OracleTimeStamp.
- *value2*
An OracleIntervalDS instance.

Return Value

An OracleTimeStamp structure.

Remarks

If either parameter has a null value, the returned OracleTimeStamp has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

operator - (OracleTimeStamp, OracleIntervalYM)

This static operator subtracts the supplied OracleIntervalYM value, from the supplied OracleTimeStamp value, and returns a new OracleTimeStamp structure.

Declaration

```
// C#  
public static operator - (OracleTimeStamp value1, OracleIntervalYM value2);
```

Parameters

- *value1*
An OracleTimeStamp.
- *value2*
An OracleIntervalYM instance.

Return Value

An OracleTimeStamp structure.

Remarks

If either parameter has a null value, the returned OracleTimeStamp has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

operator - (OracleTimeStamp, TimeSpan)

This static operator subtracts the supplied TimeSpan value, from the supplied OracleTimeStamp value, and returns a new OracleTimeStamp structure.

Declaration

```
// C#  
public static operator - (OracleTimeStamp value1, TimeSpan value2);
```

Parameters

- *value1*
An OracleTimeStamp.
- *value2*
A TimeSpan instance.

Return Value

An OracleTimeStamp structure.

Remarks

If the OracleTimeStamp instance has a null value, the returned OracleTimeStamp structure has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

OracleTimeStamp Static Type Conversions

The OracleTimeStamp static type conversions are listed in [Table 14-111](#).

Table 14-111 OracleTimeStamp Static Type Conversions

Operator	Description
explicit operator OracleTimeStamp	Converts an instance value to an OracleTimeStamp structure (Overloaded)
implicit operator OracleTimeStamp	Converts an instance value to an OracleTimeStamp structure (Overloaded)
explicit operator DateTime	Converts an OracleTimeStamp value to a DateTime structure

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

explicit operator OracleTimeStamp

explicit operator OracleTimeStamp converts the supplied value to an OracleTimeStamp structure

Overload List:

- [explicit operator OracleTimeStamp\(OracleTimeStampLTZ\)](#)
This static type conversion operator converts an OracleTimeStampLTZ value to an OracleTimeStamp structure.
- [explicit operator OracleTimeStamp\(OracleTimeStampTZ\)](#)
This static type conversion operator converts an OracleTimeStampTZ value to an OracleTimeStamp structure.
- [explicit operator OracleTimeStamp\(string\)](#)
This static type conversion operator converts the supplied string to an OracleTimeStamp structure.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

explicit operator OracleTimeStamp(OracleTimeStampLTZ)

This static type conversion operator converts an OracleTimeStampLTZ value to an OracleTimeStamp structure.

Declaration

```
// C#  
public static explicit operator OracleTimeStamp(OracleTimeStampLTZ value1);
```

Parameters

- *value1*
An OracleTimeStampLTZ instance.

Return Value

The returned `OracleTimeStamp` contains the date and time of the `OracleTimeStampLTZ` structure.

Remarks

If the `OracleTimeStampLTZ` structure has a null value, the returned `OracleTimeStamp` structure also has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

explicit operator OracleTimeStamp(OracleTimeStampTZ)

This static type conversion operator converts an `OracleTimeStampTZ` value to an `OracleTimeStamp` structure.

Declaration

```
// C#  
public static explicit operator OracleTimeStamp(OracleTimeStampTZ value1);
```

Parameters

- *value1*
An `OracleTimeStampTZ` instance.

Return Value

The returned `OracleTimeStamp` contains the date and time information from *value1*, but the time zone information from *value1* is truncated.

Remarks

If the `OracleTimeStampTZ` structure has a null value, the returned `OracleTimeStamp` structure also has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

explicit operator OracleTimeStamp(string)

This static type conversion operator converts the supplied string to an OracleTimeStamp structure.

Declaration

```
// C#  
public static explicit operator OracleTimeStamp(string tsStr);
```

Parameters

- *tsStr*
A string representation of an Oracle TIMESTAMP.

Return Value

An OracleTimeStamp.

Exceptions

ArgumentException - The *tsStr* is an invalid string representation of an Oracle TIMESTAMP or the *tsStr* is not in the timestamp format specified by the thread's OracleGlobalization.TimeStampFormat property, which represents the Oracle NLS_TIMESTAMP_FORMAT parameter.

Remarks

The names and abbreviations used for months and days are in the language specified by the DateLanguage and Calendar properties of the thread's OracleGlobalization object. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Types;  
using Oracle.DataAccess.Client;  
  
class OracleTimeStampSample  
{  
    static void Main()  
    {  
        // Set the nls_timestamp_format for the explicit  
        // operator OracleTimeStamp(string)  
        OracleGlobalization info = OracleGlobalization.GetClientInfo();  
        info.TimeStampFormat = "DD-MON-YYYY HH:MI:SS.FF AM";  
        OracleGlobalization.SetThreadInfo(info);  
  
        // construct OracleTimeStamp from a string using the format specified.  
        OracleTimeStamp ts = new OracleTimeStamp("11-NOV-1999 11:02:33.444 AM");  
  
        // Set the nls_timestamp_format for the ToString method  
        info.TimeStampFormat = "YYYY-MON-DD HH:MI:SS.FF AM";  
        OracleGlobalization.SetThreadInfo(info);  
  
        // Prints "1999-NOV-11 11:02:33.444000000 AM"
```

```
        Console.WriteLine(ts.ToString());  
    }  
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)
- *Oracle AI Database SQL Language Reference* for further information on datetime format elements

implicit operator OracleTimeStamp

This static type conversion operator converts a value to an OracleTimeStamp structure.

Overload List:

- [implicit operator OracleTimeStamp\(OracleDate\)](#)
This static type conversion operator converts an OracleDate value to an OracleTimeStamp structure.
- [implicit operator OracleTimeStamp\(DateTime\)](#)
This static type conversion operator converts a DateTime value to an OracleTimeStamp structure.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

implicit operator OracleTimeStamp(OracleDate)

This static type conversion operator converts an OracleDate value to an OracleTimeStamp structure.

Declaration

```
// C#  
public static implicit operator OracleTimeStamp (OracleDate value1);
```

Parameters

- *value1*
An OracleDate instance.

Return Value

An OracleTimeStamp structure that contains the date and time of the OracleDate structure, *value1*.

Remarks

If the OracleDate structure has a null value, the returned OracleTimeStamp structure also has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

implicit operator OracleTimeStamp(DateTime)

This static type conversion operator converts a DateTime value to an OracleTimeStamp structure.

Declaration

```
// C#  
public static implicit operator OracleTimeStamp(DateTime value);
```

Parameters

- *value*
A DateTime instance.

Return Value

An OracleTimeStamp structure.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

explicit operator DateTime

This static type conversion operator converts an `OracleTimeStamp` value to a `DateTime` structure.

Declaration

```
// C#  
public static explicit operator DateTime(OracleTimeStamp value1);
```

Parameters

- `value1`
An `OracleTimeStamp` instance.

Return Value

A `DateTime` containing the date and time in the current instance.

Exceptions

`OracleNullValueException` - The `OracleTimeStamp` structure has a null value.

Remarks

The precision of the `OracleTimeStamp` can be lost during the conversion.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

OracleTimeStamp Properties

The `OracleTimeStamp` properties are listed in [Table 14-112](#).

Table 14-112 OracleTimeStamp Properties

Properties	Description
BinData	Returns an array of bytes that represents an Oracle TIMESTAMP in Oracle internal format
Day	Specifies the day component of an <code>OracleTimeStamp</code>
IsNull	Indicates whether or not the <code>OracleTimeStamp</code> instance has a null value
Hour	Specifies the hour component of an <code>OracleTimeStamp</code>
Millisecond	Specifies the millisecond component of an <code>OracleTimeStamp</code>
Minute	Specifies the minute component of an <code>OracleTimeStamp</code>
Month	Specifies the month component of an <code>OracleTimeStamp</code>

Table 14-112 (Cont.) OracleTimeStamp Properties

Properties	Description
Nanosecond	Specifies the nanosecond component of an OracleTimeStamp
Second	Specifies the second component of an OracleTimeStamp
Value	Specifies the date and time that is stored in the OracleTimeStamp structure
Year	Specifies the year component of an OracleTimeStamp

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

BinData

This property returns an array of bytes that represents an Oracle TIMESTAMP in Oracle internal format.

Declaration

```
// C#  
public byte[] BinData {get;}
```

Property Value

A byte array that represents an Oracle TIMESTAMP in an internal format.

Exceptions

OracleNullValueException - The current instance has a null value.

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

Day

This property specifies the day component of an OracleTimeStamp.

Declaration

```
// C#  
public int Day{get;}
```


Property Value

A number that represents the day. Range of Day is (1 to 31).

Exceptions

OracleNullValueException - The current instance has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

IsNull

This property indicates whether or not the current instance has a null value.

Declaration

```
// C#  
public bool IsNull{get;}
```

Property Value

Returns true if the current instance has a null value; otherwise, returns false.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

Hour

This property specifies the hour component of an OracleTimeStamp.

Declaration

```
// C#  
public int Hour{get;}
```

Property Value

A number that represents the hour. Range of hour is (0 to 23).

Exceptions

OracleNullValueException - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

Millisecond

This property gets the millisecond component of an `OracleTimeStamp`.

Declaration

```
// C#  
public double Millisecond{get;}
```

Property Value

A number that represents a millisecond. Range of `Millisecond` is (0 to 999.999999).

Exceptions

`OracleNullValueException` - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

Minute

This property gets the minute component of an `OracleTimeStamp`.

Declaration

```
// C#  
public int Minute{get;}
```

Property Value

A number that represent a minute. Range of `Minute` is (0 to 59).

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

Month

This property gets the month component of an `OracleTimeStamp`.

Declaration

```
// C#  
public int Month{get;}
```

Property Value

A number that represents a month. Range of `Month` is (1 to 12).

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

Nanosecond

This property gets the nanosecond component of an `OracleTimeStamp`.

Declaration

```
// C#  
public int Nanosecond{get;}
```

Property Value

A number that represents a nanosecond. Range of `Nanosecond` is (0 to 999999999).

Exceptions

`OracleNullValueException` - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

Second

This property gets the second component of an `OracleTimeStamp`.

Declaration

```
// C#  
public int Second{get;}
```

Property Value

A number that represents a second. Range of `Second` is (0 to 59).

Exceptions

`OracleNullValueException` - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

Value

This property specifies the date and time that is stored in the `OracleTimeStamp` structure.

Declaration

```
// C#  
public DateTime Value{get;}
```

Property Value

A `DateTime`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

Year

This property gets the year component of an OracleTimeStamp.

Declaration

```
// C#
public int Year{get;}
```

Property Value

A number that represents a year. The range of Year is (-4712 to 9999).

Exceptions

OracleNullValueException - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

OracleTimeStamp Methods

The OracleTimeStamp methods are listed in [Table 14-113](#).

Table 14-113 OracleTimeStamp Methods

Methods	Description
AddDays	Adds the supplied number of days to the current instance
AddHours	Adds the supplied number of hours to the current instance
AddMilliseconds	Adds the supplied number of milliseconds to the current instance
AddMinutes	Adds the supplied number of minutes to the current instance
AddMonths	Adds the supplied number of months to the current instance
AddNanoseconds	Adds the supplied number of nanoseconds to the current instance
AddSeconds	Adds the supplied number of seconds to the current instance
AddYears	Adds the supplied number of years to the current instance

Table 14-113 (Cont.) OracleTimeStamp Methods

Methods	Description
CompareTo	Compares the current OracleTimeStamp instance to an object, and returns an integer that represents their relative values
Equals	Determines whether or not an object has the same date and time as the current OracleTimeStamp instance (Overloaded)
GetHashCode	Returns a hash code for the OracleTimeStamp instance
GetDaysBetween	Subtracts an OracleTimeStamp value from the current instance and returns an OracleIntervalDS that represents the time difference between the supplied OracleTimeStamp and the current instance
GetYearsBetween	Subtracts value1 from the current instance and returns an OracleIntervalYM that represents the difference between value1 and the current instance using OracleIntervalYM
GetType	Inherited from System.Object
ToOracleDate	Converts the current OracleTimeStamp structure to an OracleDate structure
ToOracleTimeStampLTZ	Converts the current OracleTimeStamp structure to an OracleTimeStampLTZ structure
ToOracleTimeStampTZ	Converts the current OracleTimeStamp structure to an OracleTimeStampTZ structure
ToString	Converts the current OracleTimeStamp structure to a string

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

AddDays

This method adds the supplied number of days to the current instance.

Declaration

```
// C#
public OracleTimeStamp AddDays(double days);
```

Parameters

- *days*
The supplied number of days. Range is $(-1,000,000,000 < days < 1,000,000,000)$

Return Value

An OracleTimeStamp.

Exceptions

ArgumentOutOfRangeException - The argument value is out of the specified range.

OracleNullValueException - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

AddHours

This method adds the supplied number of hours to the current instance.

Declaration

```
// C#  
public OracleTimeStamp AddHours(double hours);
```

Parameters

- *hours*
The supplied number of hours. Range is $(-24,000,000,000 < hours < 24,000,000,000)$.

Return Value

An OracleTimeStamp.

Exceptions

ArgumentOutOfRangeException - The argument value is out of the specified range.

OracleNullValueException - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

AddMilliseconds

This method adds the supplied number of milliseconds to the current instance.

Declaration

```
// C#  
public OracleTimeStamp AddMilliseconds(double milliseconds);
```

Parameters

- *milliseconds*

The supplied number of milliseconds. Range is $(-8.64 * 10^{16} < \text{milliseconds} < 8.64 * 10^{16})$.

Return Value

An OracleTimeStamp.

Exceptions

ArgumentOutOfRangeException - The argument value is out of the specified range.

OracleNullValueException - The current instance has a null value.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

AddMinutes

This method adds the supplied number of minutes to the current instance.

Declaration

```
// C#  
public OracleTimeStamp AddMinutes(double minutes);
```

Parameters

- *minutes*

The supplied number of minutes. Range is $(-1,440,000,000,000 < \text{minutes} < 1,440,000,000,000)$.

Return Value

An OracleTimeStamp.

Exceptions

ArgumentOutOfRangeException - The argument value is out of the specified range.

OracleNullValueException - The current instance has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

AddMonths

This method adds the supplied number of months to the current instance.

Declaration

```
// C#  
public OracleTimeStamp AddMonths(long months);
```

Parameters

- *months*

The supplied number of months. Range is $(-12,000,000,000 < months < 12,000,000,000)$.

Return Value

An OracleTimeStamp.

Exceptions

ArgumentOutOfRangeException - The argument value is out of the specified range.

OracleNullValueException - The current instance has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

AddNanoseconds

This method adds the supplied number of nanoseconds to the current instance.

Declaration

```
// C#  
public OracleTimeStamp AddNanoseconds(long nanoseconds);
```

Parameters

- *nanoseconds*

The supplied number of nanoseconds.

Return Value

An OracleTimeStamp.

Exceptions

OracleNullValueException - The current instance has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

AddSeconds

This method adds the supplied number of seconds to the current instance.

Declaration

```
// C#  
public OracleTimeStamp AddSeconds(double seconds);
```

Parameters

- *seconds*

The supplied number of seconds. Range is $(-8.64 * 10^{13} < \text{seconds} < 8.64 * 10^{13})$.

Return Value

An OracleTimeStamp.

Exceptions

ArgumentOutOfRangeException - The argument value is out of the specified range.

OracleNullValueException - The current instance has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

AddYears

This method adds the supplied number of years to the current instance.

Declaration

```
// C#  
public OracleTimeStamp AddYears(int years);
```

Parameters

- *years*

The supplied number of years. Range is (-999,999,999 <= *years* <= 999,999,999)

Return Value

An OracleTimeStamp.

Exceptions

ArgumentOutOfRangeException - The argument value is out of the specified range.

OracleNullValueException - The current instance has a null value.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

CompareTo

This method compares the current OracleTimeStamp instance to an object, and returns an integer that represents their relative values.

Declaration

```
// C#  
public int CompareTo(object obj);
```

Parameters

- *obj*

The object being compared to the current OracleTimeStamp instance.

Return Value

The method returns a number that is:

Less than zero: if the current OracleTimeStamp instance value is less than that of *obj*.

Zero: if the current OracleTimeStamp instance and *obj* values are equal.

Greater than zero: if the current OracleTimeStamp instance value is greater than that of *obj*.

Implements

IComparable

Exceptions

ArgumentException - The *obj* parameter is not of type OracleTimeStamp.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between OracleTimestamps. For example, comparing an OracleTimeStamp instance with an OracleBinary instance is not allowed. When an OracleTimeStamp is compared with a different type, an ArgumentException is thrown.
- Any OracleTimeStamp that has a value is greater than an OracleTimeStamp that has a null value.
- Two OracleTimestamps that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

Equals

Overrides Object

This method determines whether or not an object has the same date and time as the current OracleTimeStamp instance.

Declaration

```
// C#  
public override bool Equals(object obj);
```

Parameters

- *obj*

The object being compared to the current OracleTimeStamp instance.

Return Value

Returns true if the *obj* is of type OracleTimeStamp and represents the same date and time; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleTimeStamp that has a value is greater than an OracleTimeStamp that has a null value.
- Two OracleTimestamps that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

GetHashCode

Overrides Object

This method returns a hash code for the OracleTimeStamp instance.

Declaration

```
// C#  
public override int GetHashCode();
```

Return Value

A number that represents the hash code.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

GetDaysBetween

This method subtracts an OracleTimeStamp value from the current instance and returns an OracleIntervalDS that represents the time difference between the supplied OracleTimeStamp structure and the current instance.

Declaration

```
// C#  
public OracleIntervalDS GetDaysBetween(OracleTimeStamp value1);
```

Parameters

- *value1*
The OracleTimeStamp value being subtracted.

Return Value

An OracleIntervalDS that represents the interval between two OracleTimeStamp values.

Remarks

If either the current instance or the parameter has a null value, the returned `OracleIntervalDS` has a null value.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

GetYearsBetween

This method subtracts an `OracleTimeStamp` value from the current instance and returns an `OracleIntervalYM` that represents the time difference between the `OracleTimeStamp` value and the current instance.

Declaration

```
// C#  
public OracleIntervalYM GetYearsBetween(OracleTimeStamp value1);
```

Parameters

- *value1*
The `OracleTimeStamp` value being subtracted.

Return Value

An `OracleIntervalYM` that represents the interval between two `OracleTimeStamp` values.

Remarks

If either the current instance or the parameter has a null value, the returned `OracleIntervalYM` has a null value.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

ToOracleDate

This method converts the current `OracleTimeStamp` structure to an `OracleDate` structure.

Declaration

```
// C#  
public OracleDate ToOracleDate();
```

Return Value

The returned `OracleDate` contains the date and time in the current instance.

Remarks

The precision of the `OracleTimeStamp` value can be lost during the conversion.

If the value of the `OracleTimeStamp` has a null value, the value of the returned `OracleDate` structure has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

ToOracleTimeStampLTZ

This method converts the current `OracleTimeStamp` structure to an `OracleTimeStampLTZ` structure.

Declaration

```
// C#  
public OracleTimeStampLTZ ToOracleTimeStampLTZ();
```

Return Value

The returned `OracleTimeStampLTZ` contains date and time in the current instance.

Remarks

If the value of the current instance has a null value, the value of the returned `OracleTimeStampLTZ` structure has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)

ToOracleTimeStampTZ

This method converts the current `OracleTimeStamp` structure to an `OracleTimeStampTZ` structure.

Declaration

```
// C#  
public OracleTimeStampTZ ToOracleTimeStampTZ();
```

Return Value

The returned `OracleTimeStampTZ` contains the date and time from the `OracleTimeStamp` and the time zone from the `OracleGlobalization.TimeZone` of the thread.

Remarks

If the value of the current instance has a null value, the value of the returned `OracleTimeStampTZ` structure has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)

ToString

Overrides `Object`

This method converts the current `OracleTimeStamp` structure to a string.

Declaration

```
// C#  
public override string ToString();
```

Return Value

A string that represents the same date and time as the current `OracleTimeStamp` structure.

Remarks

The returned value is a string representation of an `OracleTimeStamp` in the format specified by the `OracleGlobalization.TimestampFormat` property of the thread.

The names and abbreviations used for months and days are in the language specified by the `OracleGlobalization's DateLanguage` and `Calendar` properties of the thread. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#

using System;
using Oracle.DataAccess.Types;
using Oracle.DataAccess.Client;

class ToStringSample
{
    static void Main()
    {
        // Set the nls_timestamp_format for the OracleTimeStamp(string)
        // constructor
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.TimestampFormat = "DD-MON-YYYY HH:MI:SS.FF AM";
        OracleGlobalization.SetThreadInfo(info);

        // construct OracleTimeStamp from a string using the format specified.
        OracleTimeStamp ts = new OracleTimeStamp("11-NOV-1999 11:02:33.444 AM");

        // Set the nls_timestamp_format for the ToString() method
        info.TimestampFormat = "YYYY-MON-DD HH:MI:SS.FF AM";
        OracleGlobalization.SetThreadInfo(info);

        // Prints "1999-NOV-11 11:02:33.444000000 AM"
        Console.WriteLine(ts.ToString());
    }
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStamp Structure](#)
- [OracleTimeStamp Members](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)

OracleTimeStampLTZ Structure

The `OracleTimeStampLTZ` structure represents the Oracle `TIMESTAMP WITH LOCAL TIME ZONE` data type to be stored in or retrieved from a database. Each `OracleTimeStampLTZ` stores the following information: year, month, day, hour, minute, second, and nanosecond.

Class Inheritance

`System.Object`

`System.ValueType`

`Oracle.DataAccess.Types.OracleTimeStampLTZ`

Declaration

```
// C#
public struct OracleTimeStampLTZ : IComparable, INullable, IXmlSerializable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#

using System;
using Oracle.DataAccess.Types;
using Oracle.DataAccess.Client;

class OracleTimeStampLTZSample
{
    static void Main()
    {
        // Illustrates usage of OracleTimeStampLTZ
        // Display Local Time Zone Name
        Console.WriteLine("Local Time Zone Name = " +
            OracleTimeStampLTZ.GetLocalTimeZoneName());
        OracleTimeStampLTZ tsLocal1 = OracleTimeStampLTZ.GetSysDate();
        OracleTimeStampLTZ tsLocal2 = DateTime.Now;

        // Calculate the difference between tsLocal1 and tsLocal2
        OracleIntervalDS idsDiff = tsLocal2.GetDaysBetween(tsLocal1);

        // Calculate the difference using AddNanoseconds()
        int nanoDiff = 0;
        while (tsLocal2 > tsLocal1)
        {
            nanoDiff += 10;
            tsLocal1 = tsLocal1.AddNanoseconds(10);
        }
        Console.WriteLine("idsDiff.Nanoseconds = " + idsDiff.Nanoseconds);
        Console.WriteLine("nanoDiff = " + nanoDiff);
    }
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Members](#)
- [OracleTimeStampLTZ Constructors](#)
- [OracleTimeStampLTZ Static Fields](#)
- [OracleTimeStampLTZ Static Methods](#)
- [OracleTimeStampLTZ Static Operators](#)
- [OracleTimeStampLTZ Static Type Conversions](#)
- [OracleTimeStampLTZ Properties](#)
- [OracleTimeStampLTZ Methods](#)

OracleTimeStampLTZ Members

OracleTimeStampLTZ members are listed in the following tables:

OracleTimeStampLTZ Constructors

OracleTimeStampLTZ constructors are listed in [Table 14-114](#)

Table 14-114 OracleTimeStampLTZConstructors

Constructor	Description
OracleTimeStampLTZ Constructors	Instantiates a new instance of OracleTimeStampLTZ structure (Overloaded)

OracleTimeStampLTZ Static Fields

The OracleTimeStampLTZ static fields are listed in [Table 14-115](#).

Table 14-115 OracleTimeStampLTZ Static Fields

Field	Description
MaxValue	Represents the maximum valid date for an OracleTimeStampLTZ structure, which is December 31, 9999 23:59:59.999999999
MinValue	Represents the minimum valid date for an OracleTimeStampLTZ structure, which is January 1, -4712 0:0:0
Null	Represents a null value that can be assigned to an instance of the OracleTimeStampLTZ structure

OracleTimeStampLTZ Static Methods

The OracleTimeStampLTZ static methods are listed in [Table 14-116](#).

Table 14-116 OracleTimeStampLTZ Static Methods

Methods	Description
Equals	Determines if two OracleTimeStampLTZ values are equal (Overloaded)
GetLocalTimeZoneName	Gets the client's local time zone name
GetLocalTimeZoneOffset	Gets the client's local time zone offset relative to UTC
GetSysDate	Gets an OracleTimeStampLTZ structure that represents the current date and time
GreaterThan	Determines if the first of two OracleTimeStampLTZ values is greater than the second
GreaterThanOrEqual	Determines if the first of two OracleTimeStampLTZ values is greater than or equal to the second
LessThan	Determines if the first of two OracleTimeStampLTZ values is less than the second
LessThanOrEqual	Determines if the first of two OracleTimeStampLTZ values is less than or equal to the second
NotEquals	Determines if two OracleTimeStampLTZ values are not equal
Parse	Gets an OracleTimeStampLTZ structure and sets its value for date and time using the supplied string
SetPrecision	Returns a new instance of an OracleTimeStampLTZ with the specified fractional second precision

OracleTimeStampLTZ Static Operators

The OracleTimeStampLTZ static operators are listed in [Table 14-117](#).

Table 14-117 OracleTimeStampLTZ Static Operators

Operator	Description
operator +	Adds the supplied instance value to the supplied OracleTimeStampLTZ and returns a new OracleTimeStampLTZ structure (Overloaded)
operator ==	Determines if two OracleTimeStampLTZ values are equal
operator >	Determines if the first of two OracleTimeStampLTZ values is greater than the second
operator >=	Determines if the first of two OracleTimeStampLTZ values is greater than or equal to the second
operator !=	Determines if two OracleTimeStampLTZ values are not equal
operator <	Determines if the first of two OracleTimeStampLTZ values is less than the second
operator <=	Determines if the first of two OracleTimeStampLTZ values is less than or equal to the second

Table 14-117 (Cont.) OracleTimeStampLTZ Static Operators

Operator	Description
operator -	Subtracts the supplied instance value from the supplied OracleTimeStampLTZ and returns a new OracleTimeStampLTZ structure (Overloaded)

OracleTimeStampLTZ Static Type Conversions

The OracleTimeStampLTZ static type conversions are listed in [Table 14-118](#).

Table 14-118 OracleTimeStampLTZ Static Type Conversions

Operator	Description
explicit operator OracleTimeStampLTZ	Converts an instance value to an OracleTimeStampLTZ structure (Overloaded)
implicit operator OracleTimeStampLTZ	Converts an instance value to an OracleTimeStampLTZ structure (Overloaded)
explicit operator DateTime	Converts an OracleTimeStampLTZ value to a DateTime structure

OracleTimeStampLTZ Properties

The OracleTimeStampLTZ properties are listed in [Table 14-119](#).

Table 14-119 OracleTimeStampLTZ Properties

Properties	Description
BinData	Returns an array of bytes that represents an Oracle TIMESTAMP WITH LOCAL TIME ZONE in Oracle internal format
Day	Specifies the day component of an OracleTimeStampLTZ
IsNull	Indicates whether or not the OracleTimeStampLTZ instance has a null value
Hour	Specifies the hour component of an OracleTimeStampLTZ
Millisecond	Specifies the millisecond component of an OracleTimeStampLTZ
Minute	Specifies the minute component of an OracleTimeStampLTZ
Month	Specifies the month component of an OracleTimeStampLTZ
Nanosecond	Specifies the nanosecond component of an OracleTimeStampLTZ
Second	Specifies the second component of an OracleTimeStampLTZ
Value	Specifies the date and time that is stored in the OracleTimeStampLTZ structure

Table 14-119 (Cont.) OracleTimeStampLTZ Properties

Properties	Description
Year	Specifies the year component of an OracleTimeStampLTZ

OracleTimeStampLTZ Methods

The OracleTimeStampLTZ methods are listed in [Table 14-120](#).

Table 14-120 OracleTimeStampLTZ Methods

Methods	Description
AddDays	Adds the supplied number of days to the current instance
AddHours	Adds the supplied number of hours to the current instance
AddMilliseconds	Adds the supplied number of milliseconds to the current instance
AddMinutes	Adds the supplied number of minutes to the current instance
AddMonths	Adds the supplied number of months to the current instance
AddNanoseconds	Adds the supplied number of nanoseconds to the current instance
AddSeconds	Adds the supplied number of seconds to the current instance
AddYears	Adds the supplied number of years to the current instance
CompareTo	Compares the current OracleTimeStampLTZ instance to an object and returns an integer that represents their relative values
Equals	Determines whether or not an object has the same date and time as the current OracleTimeStampLTZ instance (Overloaded)
GetHashCode	Returns a hash code for the OracleTimeStampLTZ instance
GetDaysBetween	Subtracts an OracleTimeStampLTZ from the current instance and returns an OracleIntervalDS that represents the difference
GetYearsBetween	Subtracts an OracleTimeStampLTZ from the current instance and returns an OracleIntervalYM that represents the difference
GetType	Inherited from System.Object
ToOracleDate	Converts the current OracleTimeStampLTZ structure to an OracleDate structure
ToOracleTimeStamp	Converts the current OracleTimeStampLTZ structure to an OracleTimeStamp structure

Table 14-120 (Cont.) OracleTimeStampLTZ Methods

Methods	Description
ToOracleTimeStampTZ	Converts the current OracleTimeStampLTZ structure to an OracleTimeStampTZ structure
ToString	Converts the current OracleTimeStampLTZ structure to a string
ToUniversalTime	Converts the current local time to Coordinated Universal Time (UTC)

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)

OracleTimeStampLTZ Constructors

The OracleTimeStampLTZ constructors create new instances of the OracleTimeStampLTZ structure.

Overload List:

- [OracleTimeStampLTZ\(DateTime\)](#)
This constructor creates a new instance of the OracleTimeStampLTZ structure and sets its value for date and time using the supplied DateTime value.
- [OracleTimeStampLTZ\(string\)](#)
This constructor creates a new instance of the OracleTimeStampLTZ structure and sets its value for date and time using the supplied string.
- [OracleTimeStampLTZ\(int, int, int\)](#)
This constructor creates a new instance of the OracleTimeStampLTZ structure and sets its value for date using year, month, and day.
- [OracleTimeStampLTZ\(int, int, int, int, int, int\)](#)
This constructor creates a new instance of the OracleTimeStampLTZ structure and sets its value for date and time using year, month, day, hour, minute, and second.
- [OracleTimeStampLTZ\(int, int, int, int, int, int, double\)](#)
This constructor creates a new instance of the OracleTimeStampLTZ structure and sets its value for date and time using year, month, day, hour, minute, second, and millisecond.
- [OracleTimeStampLTZ\(int, int, int, int, int, int, int\)](#)
This constructor creates a new instance of the OracleTimeStampLTZ structure and sets its value for date and time using year, month, day, hour, minute, second, and nanosecond.
- [OracleTimeStampLTZ\(byte \[\]\)](#)

This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value to the provided byte array, which is in the internal Oracle `TIMESTAMP WITH LOCAL TIME ZONE` format.

📘 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

OracleTimeStampLTZ(DateTime)

This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value for date and time using the supplied `DateTime` value.

Declaration

```
// C#  
public OracleTimeStampLTZ (DateTime dt);
```

Parameters

- *dt*
The supplied `DateTime` value.

Exceptions

`ArgumentException` - The *dt* parameter cannot be used to construct a valid `OracleTimeStampLTZ`.

📘 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

OracleTimeStampLTZ(string)

This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value for date and time using the supplied string.

Declaration

```
// C#  
public OracleTimeStampLTZ(string tsStr);
```

Parameters

- *tsStr*

A string that represents an Oracle TIMESTAMP WITH LOCAL TIME ZONE.

Exceptions

ArgumentException - The *tsStr* is an invalid string representation of an Oracle TIMESTAMP WITH LOCAL TIME ZONE or the supplied *tsStr* is not in the timestamp format specified by the `OracleGlobalization.TimeStampFormat` property of the thread, which represents the Oracle `NLS_TIMESTAMP_FORMAT` parameter.

ArgumentNullException - The *tsStr* value is null.

Remarks

The names and abbreviations used for months and days are in the language specified by the `DateLanguage` and `Calendar` properties of the thread's `OracleGlobalization` object. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class OracleTimeStampLTZSample
{
    static void Main()
    {
        // Set the nls_timestamp_format for the OracleTimeStampLTZ(string)
        // constructor
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.TimeStampFormat = "DD-MON-YYYY HH:MI:SS.FF AM";
        OracleGlobalization.SetThreadInfo(info);

        // construct OracleTimeStampLTZ from a string using the format
        // specified.
        OracleTimeStampLTZ ts =
            new OracleTimeStampLTZ("11-NOV-1999 11:02:33.444 AM");

        // Set the nls_timestamp_format for the ToString() method
        info.TimeStampFormat = "YYYY-MON-DD HH:MI:SS.FF AM";
        OracleGlobalization.SetThreadInfo(info);

        // Prints "1999-NOV-11 11:02:33.444000000 AM"
        Console.WriteLine(ts.ToString());
    }
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)
- *Oracle AI Database SQL Language Reference* for further information on date format elements

OracleTimeStampLTZ(int, int, int)

This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value for date using year, month, and day.

Declaration

```
// C#  
public OracleTimeStampLTZ(int year, int month, int day);
```

Parameters

- *year*
The year provided. Range of *year* is (-4712 to 9999).
- *month*
The month provided. Range of *month* is (1 to 12).
- *day*
The day provided. Range of *day* is (1 to 31).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampLTZ` (that is, the day is out of range for the month).

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

OracleTimeStampLTZ(int, int, int, int, int, int)

This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value for date and time using year, month, day, hour, minute, and second.

Declaration

```
// C#  
public OracleTimeStampLTZ (int year, int month, int day, int hour,  
    int minute, int second);
```

Parameters

- *year*
The year provided. Range of *year* is (-4712 to 9999).
- *month*
The month provided. Range of *month* is (1 to 12).
- *day*
The day provided. Range of *day* is (1 to 31).
- *hour*
The hour provided. Range of *hour* is (0 to 23).
- *minute*
The minute provided. Range of *minute* is (0 to 59).
- *second*
The second provided. Range of *second* is (0 to 59).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampLTZ` (that is, the day is out of range for the month).

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

OracleTimeStampLTZ(int, int, int, int, int, int, double)

This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, and millisecond.

Declaration

```
// C#  
public OracleTimeStampLTZ(int year, int month, int day, int hour, int minute, int  
second, double millisecond);
```

Parameters

- *year*
The year provided. Range of *year* is (-4712 to 9999).
- *month*
The month provided. Range of *month* is (1 to 12).
- *day*
The day provided. Range of *day* is (1 to 31).
- *hour*
The hour provided. Range of *hour* is (0 to 23).
- *minute*
The minute provided. Range of *minute* is (0 to 59).
- *second*
The second provided. Range of *second* is (0 to 59).
- *milliSeconds*
The milliseconds provided. Range of *millisecond* is (0 to 999.999999).

Exceptions

ArgumentOutOfRangeException - The argument value for one or more of the parameters is out of the specified range.

ArgumentException - The argument values of the parameters cannot be used to construct a valid OracleTimeStampLTZ (that is, the day is out of range for the month).

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

OracleTimeStampLTZ(int, int, int, int, int, int, int)

This constructor creates a new instance of the OracleTimeStampLTZ structure and sets its value for date and time using year, month, day, hour, minute, second, and nanosecond.

Declaration

```
// C#  
public OracleTimeStampLTZ (int year, int month, int day, int hour,  
    int minute, int second, int nanosecond);
```

Parameters

- *year*
The year provided. Range of *year* is (-4712 to 9999).
- *month*
The month provided. Range of *month* is (1 to 12).
- *day*
The day provided. Range of *day* is (1 to 31).
- *hour*
The hour provided. Range of *hour* is (0 to 23).
- *minute*
The minute provided. Range of *minute* is (0 to 59).
- *second*
The second provided. Range of *second* is (0 to 59).
- *nanosecond*
The nanosecond provided. Range of *nanosecond* is (0 to 999999999).

Exceptions

ArgumentOutOfRangeException - The argument value for one or more of the parameters is out of the specified range.

ArgumentException - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampLTZ` (that is, the day is out of range for the month).

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

OracleTimeStampLTZ(byte [])

This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value to the provided byte array, which is in the internal Oracle `TIMESTAMP WITH LOCAL TIME ZONE` format.

Declaration

```
// C#
public OracleTimeStampLTZ (byte[] bytes);
```

Parameters

- bytes*
A byte array that represents an Oracle TIMESTAMP WITH LOCAL TIME ZONE in Oracle internal format.

Exceptions

ArgumentException - *bytes* is not in an internal Oracle TIMESTAMP WITH LOCAL TIME ZONE format or *bytes* is not a valid Oracle TIMESTAMP WITH LOCAL TIME ZONE.

ArgumentNullException - *bytes* is null.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

OracleTimeStampLTZ Static Fields

The OracleTimeStampLTZ static fields are listed in [Table 14-121](#).

Table 14-121 OracleTimeStampLTZ Static Fields

Field	Description
MaxValue	Represents the maximum valid date for an OracleTimeStampLTZ structure, which is December 31, 9999 23:59:59.999999999
MinValue	Represents the minimum valid date for an OracleTimeStampLTZ structure, which is January 1, -4712 0:0:0
Null	Represents a null value that can be assigned to an instance of the OracleTimeStampLTZ structure

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

MaxValue

This static field represents the maximum valid date for an OracleTimeStampLTZ structure, which is December 31, 9999 23:59:59.999999999.

Declaration

```
// C#  
public static readonly OracleTimeStampLTZ MaxValue;
```

Remarks

This value is the maximum date and time in the client time zone.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

MinValue

This static field represents the minimum valid date for an OracleTimeStampLTZ structure, which is January 1, -4712 0:0:0.

Declaration

```
// C#  
public static readonly OracleTimeStampLTZ MinValue;
```

Remarks

This value is the minimum date and time in the client time zone.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

Null

This static field represents a null value that can be assigned to an instance of the OracleTimeStampLTZ structure.

Declaration

```
// C#  
public static readonly OracleTimeStampLTZ Null;
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

OracleTimeStampLTZ Static Methods

The OracleTimeStampLTZ static methods are listed in [Table 14-122](#).

Table 14-122 OracleTimeStampLTZ Static Methods

Methods	Description
Equals	Determines if two OracleTimeStampLTZ values are equal (Overloaded)
GetLocalTimeZoneName	Gets the client's local time zone name
GetLocalTimeZoneOffset	Gets the client's local time zone offset relative to UTC
GetSysDate	Gets an OracleTimeStampLTZ structure that represents the current date and time
GreaterThan	Determines if the first of two OracleTimeStampLTZ values is greater than the second
GreaterThanOrEqual	Determines if the first of two OracleTimeStampLTZ values is greater than or equal to the second
LessThan	Determines if the first of two OracleTimeStampLTZ values is less than the second
LessThanOrEqual	Determines if the first of two OracleTimeStampLTZ values is less than or equal to the second
NotEquals	Determines if two OracleTimeStampLTZ values are not equal
Parse	Gets an OracleTimeStampLTZ structure and sets its value for date and time using the supplied string
SetPrecision	Returns a new instance of an OracleTimeStampLTZ with the specified fractional second precision

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

Equals

This static method determines if two OracleTimeStampLTZ values are equal.

Declaration

```
// C#  
public static bool Equals(OracleTimeStampLTZ value1,  
    OracleTimeStampLTZ value2);
```

Parameters

- *value1*
The first OracleTimeStampLTZ.
- *value2*
The second OracleTimeStampLTZ.

Return Value

Returns true if two OracleTimeStampLTZ values are equal. Returns false otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any OracleTimeStampLTZ that has a value is greater than an OracleTimeStampLTZ that has a null value.
- Two OracleTimeStampLTZs that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

GetLocalTimeZoneName

This static method gets the client's local time zone name.

Declaration

```
// C#  
public static string GetLocalTimeZoneName();
```

Return Value

A string containing the local time zone.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

GetLocalTimeZoneOffset

This static method gets the client's local time zone offset relative to Coordinated Universal Time (UTC).

Declaration

```
// C#  
public static TimeSpan GetLocalTimeZoneOffset( );
```

Return Value

A TimeSpan structure containing the local time zone hours and time zone minutes.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

GetSysDate

This static method gets an OracleTimeStampLTZ structure that represents the current date and time.

Declaration

```
// C#  
public static OracleTimeStampLTZ GetSysDate();
```

Return Value

An OracleTimeStampLTZ structure that represents the current date and time.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

GreaterThan

This static method determines if the first of two `OracleTimeStampLTZ` values is greater than the second.

Declaration

```
// C#  
public static bool GreaterThan(OracleTimeStampLTZ value1,  
    OracleTimeStampLTZ value2);
```

Parameters

- *value1*
The first `OracleTimeStampLTZ`.
- *value2*
The second `OracleTimeStampLTZ`.

Return Value

Returns `true` if the first of two `OracleTimeStampLTZ` values is greater than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampLTZ` that has a value is greater than an `OracleTimeStampLTZ` that has a null value.
- Two `OracleTimeStampLTZ`s that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

GreaterThanOrEqual

This static method determines if the first of two `OracleTimeStampLTZ` values is greater than or equal to the second.

Declaration

```
// C#  
public static bool GreaterThanOrEqual(OracleTimeStampLTZ value1,  
    OracleTimeStampLTZ value2);
```

Parameters

- *value1*

The first `OracleTimeStampLTZ`.

- `value2`

The second `OracleTimeStampLTZ`.

Return Value

Returns `true` if the first of two `OracleTimeStampLTZ` values is greater than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampLTZ` that has a value is greater than an `OracleTimeStampLTZ` that has a null value.
- Two `OracleTimeStampLTZ`s that contain a null value are equal.

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

LessThan

This static method determines if the first of two `OracleTimeStampLTZ` values is less than the second.

Declaration

```
// C#  
public static bool LessThan(OracleTimeStampLTZ value1,  
    OracleTimeStampLTZ value2);
```

Parameters

- `value1`
The first `OracleTimeStampLTZ`.
- `value2`
The second `OracleTimeStampLTZ`.

Return Value

Returns `true` if the first of two `OracleTimeStampLTZ` values is less than the second. Returns `false` otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampLTZ` that has a value is greater than an `OracleTimeStampLTZ` that has a null value.

- Two OracleTimeStampLTZs that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

LessThanOrEqual

This static method determines if the first of two OracleTimeStampLTZ values is less than or equal to the second.

Declaration

```
// C#  
public static bool LessThanOrEqual(OracleTimeStampLTZ value1,  
    OracleTimeStampLTZ value2);
```

Parameters

- *value1*
The first OracleTimeStampLTZ.
- *value2*
The second OracleTimeStampLTZ.

Return Value

Returns true if the first of two OracleTimeStampLTZ values is less than or equal to the second. Returns false otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any OracleTimeStampLTZ that has a value is greater than an OracleTimeStampLTZ that has a null value.
- Two OracleTimeStampLTZs that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

NotEquals

This static method determines if two OracleTimeStampLTZ values are not equal.

Declaration

```
// C#  
public static bool NotEquals(OracleTimeStampLTZ value1,  
    OracleTimeStampLTZ value2);
```

Parameters

- *value1*
The first OracleTimeStampLTZ.
- *value2*
The second OracleTimeStampLTZ.

Return Value

Returns true if two OracleTimeStampLTZ values are not equal. Returns false otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any OracleTimeStampLTZ that has a value is greater than an OracleTimeStampLTZ that has a null value.
- Two OracleTimeStampLTZs that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

Parse

This static method creates an OracleTimeStampLTZ structure and sets its value using the supplied string.

Declaration

```
// C#  
public static OracleTimeStampLTZ Parse(string tsStr);
```

Parameters

- *tsStr*
A string that represents an Oracle TIMESTAMP WITH LOCAL TIME ZONE.

Return Value

An OracleTimeStampLTZ structure.

Exceptions

ArgumentException - The *tsStr* parameter is an invalid string representation of an Oracle **TIMESTAMP WITH LOCAL TIME ZONE** or the *tsStr* is not in the timestamp format specified by the `OracleGlobalization.TimeStampFormat` property of the thread, which represents the Oracle `NLS_TIMESTAMP_FORMAT` parameter.

ArgumentNullException - The *tsStr* value is null.

Remarks

The names and abbreviations used for months and days are in the language specified by the `DateLanguage` and `Calendar` properties of the thread's `OracleGlobalization` object. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#

using System;
using Oracle.DataAccess.Types;
using Oracle.DataAccess.Client;

class ParseSample
{
    static void Main()
    {
        // Set the nls_timestamp_format for the Parse() method
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.TimeStampFormat = "DD-MON-YYYY HH:MI:SS.FF AM";
        OracleGlobalization.SetThreadInfo(info);

        // construct OracleTimeStampLTZ from a string using the format specified.
        OracleTimeStampLTZ ts =
            OracleTimeStampLTZ.Parse("11-NOV-1999 11:02:33.444 AM");

        // Set the nls_timestamp_format for the ToString() method
        info.TimeStampFormat = "YYYY-MON-DD HH:MI:SS.FF AM";
        OracleGlobalization.SetThreadInfo(info);

        // Prints "1999-NOV-11 11:02:33.444000000 AM"
        Console.WriteLine(ts.ToString());
    }
}
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)

SetPrecision

This static method returns a new instance of an `OracleTimeStampLTZ` with the specified fractional second precision.

Declaration

```
// C#  
public static OracleTimeStampLTZ SetPrecision(OracleTimeStampLTZ value1,  
    int fracSecPrecision);
```

Parameters

- *value1*
The provided `OracleTimeStampLTZ` object.
- *fracSecPrecision*
The fractional second precision provided. Range of fractional second precision is (0 to 9).

Return Value

An `OracleTimeStampLTZ` structure with the specified fractional second precision

Exceptions

`ArgumentOutOfRangeException` - *fracSecPrecision* is out of the specified range.

Remarks

The value specified in the supplied *fracSecPrecision* parameter is used to perform a rounding off operation on the supplied `OracleTimeStampLTZ` value. Depending on this value, 0 or more trailing zeros are displayed in the string returned by `ToString()`.

Example

The `OracleTimeStampLTZ` with a value of "December 31, 9999 23:59:59.99" results in the string "December 31, 9999 23:59:59.99000" when `SetPrecision()` is called with the fractional second precision set to 5.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

OracleTimeStampLTZ Static Operators

The `OracleTimeStampLTZ` static operators are listed in [Table 14-123](#).

Table 14-123 OracleTimeStampLTZ Static Operators

Operator	Description
operator +	Adds the supplied instance value to the supplied OracleTimeStampLTZ and returns a new OracleTimeStampLTZ structure (Overloaded)
operator ==	Determines if two OracleTimeStampLTZ values are equal
operator >	Determines if the first of two OracleTimeStampLTZ values is greater than the second
operator >=	Determines if the first of two OracleTimeStampLTZ values is greater than or equal to the second
operator !=	Determines if two OracleTimeStampLTZ values are not equal
operator <	Determines if the first of two OracleTimeStampLTZ values is less than the second
operator <=	Determines if the first of two OracleTimeStampLTZ values is less than or equal to the second
operator -	Subtracts the supplied instance value from the supplied OracleTimeStampLTZ and returns a new OracleTimeStampLTZ structure (Overloaded)

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

operator +

operator + adds the supplied value to the supplied OracleTimeStampLTZ and returns a new OracleTimeStampLTZ structure.

Overload List:

- [operator + \(OracleTimeStampLTZ, OracleIntervalDS\)](#)
This static operator adds the supplied OracleIntervalDS to the supplied OracleTimeStampLTZ and returns a new OracleTimeStampLTZ structure.
- [operator + \(OracleTimeStampLTZ, OracleIntervalYM\)](#)
This static operator adds the supplied OracleIntervalYM to the supplied OracleTimeStampLTZ and returns a new OracleTimeStampLTZ structure.
- [operator + \(OracleTimeStampLTZ, TimeSpan\)](#)
This static operator adds the supplied TimeSpan to the supplied OracleTimeStampLTZ and returns a new OracleTimeStampLTZ structure.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

operator + (OracleTimeStampLTZ, OracleIntervalDS)

This static operator adds the supplied `OracleIntervalDS` to the supplied `OracleTimeStampLTZ` and returns a new `OracleTimeStampLTZ` structure.

Declaration

```
// C#  
public static operator +(OracleTimeStampLTZ value1,  
    OracleIntervalDS value2);
```

Parameters

- `value1`
An `OracleTimeStampLTZ`.
- `value2`
An `OracleIntervalDS`.

Return Value

An `OracleTimeStampLTZ`.

Remarks

If either parameter has a null value, the returned `OracleTimeStampLTZ` has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

operator + (OracleTimeStampLTZ, OracleIntervalYM)

This static operator adds the supplied `OracleIntervalYM` to the supplied `OracleTimeStampLTZ` and returns a new `OracleTimeStampLTZ` structure.

Declaration

```
// C#  
public static operator +(OracleTimeStampLTZ value1,  
    OracleIntervalYM value2);
```

Parameters

- *value1*
An OracleTimeStampLTZ.
- *value2*
An OracleIntervalYM.

Return Value

An OracleTimeStampLTZ.

Remarks

If either parameter has a null value, the returned OracleTimeStampLTZ has a null value.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

operator + (OracleTimeStampLTZ, TimeSpan)

This static operator adds the supplied TimeSpan to the supplied OracleTimeStampLTZ and returns a new OracleTimeStampLTZ structure.

Declaration

```
// C#  
public static operator +(OracleTimeStampLTZ value1, TimeSpan value2);
```

Parameters

- *value1*
An OracleTimeStampLTZ.
- *value2*
A TimeSpan.

Return Value

An OracleTimeStampLTZ.

Remarks

If the OracleTimeStampLTZ instance has a null value, the returned OracleTimeStampLTZ has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

operator ==

This static operator determines if two `OracleTimeStampLTZ` values are equal.

Declaration

```
// C#  
public static bool operator == (OracleTimeStampLTZ value1,  
    OracleTimeStampLTZ value2);
```

Parameters

- *value1*
The first `OracleTimeStampLTZ`.
- *value2*
The second `OracleTimeStampLTZ`.

Return Value

Returns `true` if they are the same; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampLTZ` that has a value is greater than an `OracleTimeStampLTZ` that has a null value.
- Two `OracleTimeStampLTZ`s that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

operator >

This static operator determines if the first of two `OracleTimeStampLTZ` values is greater than the second.

Declaration

```
// C#
public static bool operator > (OracleTimeStampLTZ value1,
    OracleTimeStampLTZ value2);
```

Parameters

- *value1*
The first OracleTimeStampLTZ.
- *value2*
The second OracleTimeStampLTZ.

Return Value

Returns true if the first OracleTimeStampLTZ value is greater than the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleTimeStampLTZ that has a value is greater than an OracleTimeStampLTZ that has a null value.
- Two OracleTimeStampLTZs that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

operator >=

This static operator determines if the first of two OracleTimeStampLTZ values is greater than or equal to the second.

Declaration

```
// C#
public static bool operator >= (OracleTimeStampLTZ value1,
    OracleTimeStampLTZ value2);
```

Parameters

- *value1*
An OracleTimeStampLTZ.
- *value2*
The second OracleTimeStampLTZ.

Return Value

Returns true if the first `OracleTimeStampLTZ` is greater than or equal to the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampLTZ` that has a value is greater than an `OracleTimeStampLTZ` that has a null value.
- Two `OracleTimeStampLTZ`s that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

operator !=

This static operator determines if two `OracleTimeStampLTZ` values are not equal.

Declaration

```
// C#
public static bool operator != (OracleTimeStampLTZ value1,
    OracleTimeStampLTZ value2);
```

Parameters

- *value1*
The first `OracleTimeStampLTZ`.
- *value2*
The second `OracleTimeStampLTZ`.

Return Value

Returns true if two `OracleTimeStampLTZ` values are not equal; otherwise returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampLTZ` that has a value is greater than an `OracleTimeStampLTZ` that has a null value.
- Two `OracleTimeStampLTZ`s that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

operator <

This static operator determines if the first of two `OracleTimeStampLTZ` values is less than the second.

Declaration

```
// C#  
public static bool operator < (OracleTimeStampLTZ value1,  
    OracleTimeStampLTZ value2);
```

Parameters

- *value1*
The first `OracleTimeStampLTZ`.
- *value2*
The second `OracleTimeStampLTZ`.

Return Value

Returns `true` if the first `OracleTimeStampLTZ` is less than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampLTZ` that has a value is greater than an `OracleTimeStampLTZ` that has a null value.
- Two `OracleTimeStampLTZ`s that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

operator <=

This static operator determines if the first of two `OracleTimeStampLTZ` values is less than or equal to the second.

Declaration

```
// C#  
public static bool operator <= (OracleTimeStampLTZ value1,  
    OracleTimeStampLTZ value2);
```

Parameters

- *value1*
The first OracleTimeStampLTZ.
- *value2*
The second OracleTimeStampLTZ.

Return Value

Returns true if the first OracleTimeStampLTZ is less than or equal to the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleTimeStampLTZ that has a value is greater than an OracleTimeStampLTZ that has a null value.
- Two OracleTimeStampLTZs that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

operator -

operator - subtracts the supplied value, from the supplied OracleTimeStampLTZ value, and returns a new OracleTimeStampLTZ structure.

Overload List:

- [operator - \(OracleTimeStampLTZ, OracleIntervalDS\)](#)
This static operator subtracts the supplied OracleIntervalDS value, from the supplied OracleTimeStampLTZ value, and return a new OracleTimeStampLTZ structure.
- [operator - \(OracleTimeStampLTZ, OracleIntervalYM\)](#)
This static operator subtracts the supplied OracleIntervalYM value, from the supplied OracleTimeStampLTZ value, and returns a new OracleTimeStampLTZ structure.
- [operator - \(OracleTimeStampLTZ, TimeSpan\)](#)
This static operator subtracts the supplied TimeSpan value, from the supplied OracleTimeStampLTZ value, and returns a new OracleTimeStampLTZ structure.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

operator - (OracleTimeStampLTZ, OracleIntervalDS)

This static operator subtracts the supplied `OracleIntervalDS` value, from the supplied `OracleTimeStampLTZ` value, and return a new `OracleTimeStampLTZ` structure.

Declaration

```
// C#  
public static operator - (OracleTimeStampLTZ value1,  
    OracleIntervalDS value2);
```

Parameters

- *value1*
An `OracleTimeStampLTZ`.
- *value2*
An `OracleIntervalDS` instance.

Return Value

An `OracleTimeStampLTZ` structure.

Remarks

If either parameter has a null value, the returned `OracleTimeStampLTZ` has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

operator - (OracleTimeStampLTZ, OracleIntervalYM)

This static operator subtracts the supplied `OracleIntervalYM` value, from the supplied `OracleTimeStampLTZ` value, and returns a new `OracleTimeStampLTZ` structure.

Declaration

```
// C#  
public static operator - (OracleTimeStampLTZ value1,  
    OracleIntervalYM value2);
```

Parameters

- *value1*
An OracleTimeStampLTZ.
- *value2*
An OracleIntervalYM.

Return Value

An OracleTimeStampLTZ structure.

Remarks

If either parameter has a null value, the returned OracleTimeStampLTZ has a null value.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

operator - (OracleTimeStampLTZ, TimeSpan)

This static operator subtracts the supplied TimeSpan value, from the supplied OracleTimeStampLTZ value, and returns a new OracleTimeStampLTZ structure.

Declaration

```
// C#  
public static operator -(OracleTimeStampLTZ value1, TimeSpan value2);
```

Parameters

- *value1*
An OracleTimeStampLTZ.
- *value2*
A TimeSpan.

Return Value

An OracleTimeStampLTZ structure.

Remarks

If the OracleTimeStampLTZ instance has a null value, the returned OracleTimeStampLTZ structure has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

OracleTimeStampLTZ Static Type Conversions

The OracleTimeStampLTZ static type conversions are listed in [Table 14-124](#).

Table 14-124 OracleTimeStampLTZ Static Type Conversions

Operator	Description
explicit operator OracleTimeStampLTZ	Converts an instance value to an OracleTimeStampLTZ structure (Overloaded)
implicit operator OracleTimeStampLTZ	Converts an instance value to an OracleTimeStampLTZ structure (Overloaded)
explicit operator DateTime	Converts an OracleTimeStampLTZ value to a DateTime structure

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

explicit operator OracleTimeStampLTZ

explicit operator OracleTimeStampLTZ converts the supplied value to an OracleTimeStampLTZ structure.

Overload List:

- [explicit operator OracleTimeStampLTZ\(OracleTimeStamp\)](#)
This static type conversion operator converts an OracleTimeStamp value to an OracleTimeStampLTZ structure.
- [explicit operator OracleTimeStampLTZ\(OracleTimeStampTZ\)](#)
This static type conversion operator converts an OracleTimeStampTZ value to an OracleTimeStampLTZ structure.
- [explicit operator OracleTimeStampLTZ\(string\)](#)
This static type conversion operator converts the supplied string to an OracleTimeStampLTZ structure.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

explicit operator OracleTimeStampLTZ(OracleTimeStamp)

This static type conversion operator converts an `OracleTimeStamp` value to an `OracleTimeStampLTZ` structure.

Declaration

```
// C#  
public static explicit operator OracleTimeStampLTZ (OracleTimeStamp value1);
```

Parameters

- *value1*
An `OracleTimeStamp`.

Return Value

The `OracleTimeStampLTZ` structure contains the date and time of the `OracleTimeStampTZ` structure.

Remarks

If the `OracleTimeStamp` structure has a null value, the returned `OracleTimeStampLTZ` structure also has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

explicit operator OracleTimeStampLTZ(OracleTimeStampTZ)

This static type conversion operator converts an `OracleTimeStampTZ` value to an `OracleTimeStampLTZ` structure.

Declaration

```
// C#  
public static explicit operator OracleTimeStampLTZ  
(OracleTimeStampTZ value1);
```

Parameters

- *value1*
An OracleTimeStampTZ instance.

Return Value

The OracleTimeStampLTZ structure contains the date and time in the OracleTimeStampTZ structure (which is normalized to the client local time zone).

Remarks

If the OracleTimeStampTZ structure has a null value, the returned OracleTimeStampLTZ structure also has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

explicit operator OracleTimeStampLTZ(string)

This static type conversion operator converts the supplied string to an OracleTimeStampLTZ structure.

Declaration

```
// C#  
public static explicit operator OracleTimeStampLTZ (string tsStr);
```

Parameters

- *tsStr*
A string representation of an Oracle TIMESTAMP WITH LOCAL TIME ZONE.

Return Value

A OracleTimeStampLTZ.

Exceptions

ArgumentException - The *tsStr* parameter is an invalid string representation of an Oracle TIMESTAMP WITH LOCAL TIME ZONE or the *tsStr* is not in the timestamp format specified by the thread's OracleGlobalization.TimestampFormat property, which represents the Oracle NLS_TIMESTAMP_FORMAT parameter.

Remarks

The names and abbreviations used for months and days are in the language specified by the DateLanguage and Calendar properties of the thread's OracleGlobalization object. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#

using System;
using Oracle.DataAccess.Types;
using Oracle.DataAccess.Client;

class OracleTimeStampLTZSample
{
    static void Main()
    {
        // Set the nls_timestamp_format for the OracleTimeStampLTZ(string)
        // constructor
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.TimestampFormat = "DD-MON-YYYY HH:MI:SS.FF AM";
        OracleGlobalization.SetThreadInfo(info);

        // construct OracleTimeStampLTZ from a string using the format specified.
        OracleTimeStampLTZ ts =
            new OracleTimeStampLTZ("11-NOV-1999 11:02:33.444 AM");

        // Set the nls_timestamp_format for the ToString() method
        info.TimestampFormat = "YYYY-MON-DD HH:MI:SS.FF AM";
        OracleGlobalization.SetThreadInfo(info);

        // Prints "1999-NOV-11 11:02:33.444000000 AM"
        Console.WriteLine(ts.ToString());
    }
}
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)
- *Oracle AI Database SQL Language Reference* for further information on datetime format elements

implicit operator OracleTimeStampLTZ

implicit operator OracleTimeStampLTZ converts the supplied structure to an OracleTimeStampLTZ structure.

Overload List:

- [implicit operator OracleTimeStampLTZ\(OracleDate\)](#)

This static type conversion operator converts an OracleDate value to an OracleTimeStampLTZ structure.

- [implicit operator OracleTimeStampLTZ\(DateTime\)](#)

This static type conversion operator converts a `DateTime` structure to an `OracleTimeStampLTZ` structure.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

implicit operator OracleTimeStampLTZ(OracleDate)

This static type conversion operator converts an `OracleDate` value to an `OracleTimeStampLTZ` structure.

Declaration

```
// C#  
public static implicit operator OracleTimeStampLTZ(OracleDate value1);
```

Parameters

- *value1*
An `OracleDate`.

Return Value

The returned `OracleTimeStampLTZ` structure contains the date and time in the `OracleDate` structure.

Remarks

If the `OracleDate` structure has a null value, the returned `OracleTimeStampLTZ` structure also has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

implicit operator OracleTimeStampLTZ(DateTime)

This static type conversion operator converts a `DateTime` structure to an `OracleTimeStampLTZ` structure.

Declaration

```
// C#  
public static implicit operator OracleTimeStampLTZ(DateTime value1);
```

Parameters

- *value1*
A DateTime structure.

Return Value

An OracleTimeStampLTZ structure.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

explicit operator DateTime

This static type conversion operator converts an OracleTimeStampLTZ value to a DateTime structure.

Declaration

```
// C#  
public static explicit operator DateTime(OracleTimeStampLTZ value1);
```

Parameters

- *value1*
An OracleTimeStampLTZ instance.

Return Value

A DateTime that contains the date and time in the current instance.

Exceptions

OracleNullValueException - The OracleTimeStampLTZ structure has a null value.

Remarks

The precision of the OracleTimeStampLTZ value can be lost during the conversion.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

OracleTimeStampLTZ Properties

The OracleTimeStampLTZ properties are listed in [Table 14-125](#).

Table 14-125 OracleTimeStampLTZ Properties

Properties	Description
BinData	Returns an array of bytes that represents an Oracle TIMESTAMP WITH LOCAL TIME ZONE in Oracle internal format
Day	Specifies the day component of an OracleTimeStampLTZ
IsNull	Indicates whether or not the OracleTimeStampLTZ instance has a null value
Hour	Specifies the hour component of an OracleTimeStampLTZ
Millisecond	Specifies the millisecond component of an OracleTimeStampLTZ
Minute	Specifies the minute component of an OracleTimeStampLTZ
Month	Specifies the month component of an OracleTimeStampLTZ
Nanosecond	Specifies the nanosecond component of an OracleTimeStampLTZ
Second	Specifies the second component of an OracleTimeStampLTZ
Value	Specifies the date and time that is stored in the OracleTimeStampLTZ structure
Year	Specifies the year component of an OracleTimeStampLTZ

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

BinData

This property returns an array of bytes that represents an Oracle TIMESTAMP WITH LOCAL TIME ZONE in Oracle internal format.

Declaration

```
// C#
public byte[] BinData {get;}
```

Property Value

A byte array that represents an Oracle TIMESTAMP WITH LOCAL TIME ZONE internal format.

Exceptions

OracleNullValueException - The current instance has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

Day

This property specifies the day component of an OracleTimeStampLTZ.

Declaration

```
// C#  
public int Day{get;}
```

Property Value

A number that represents the day. Range of Day is (1 to 31).

Exceptions

OracleNullValueException - The current instance has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

IsNull

This property indicates whether or not the current instance has a null value.

Declaration

```
// C#  
public bool IsNull{get;}
```

Property Value

Returns true if the current instance contains a null value; otherwise, returns false.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

Hour

This property specifies the hour component of an `OracleTimeStampLTZ`.

Declaration

```
// C#  
public int Hour{get;}
```

Property Value

A number that represents the hour. Range of Hour is (0 to 23).

Exceptions

`OracleNullValueException` - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

Millisecond

This property gets the millisecond component of an `OracleTimeStampLTZ`.

Declaration

```
// C#  
public double Millisecond{get;}
```

Property Value

A number that represents a millisecond. Range of Millisecond is (0 to 999.999999)

Exceptions

`OracleNullValueException` - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

Minute

This property gets the minute component of an `OracleTimeStampLTZ`.

Declaration

```
// C#  
public int Minute{get;}
```

Property Value

A number that represent a minute. Range of Minute is (0 to 59).

Exceptions

`OracleNullValueException` - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

Month

This property gets the month component of an `OracleTimeStampLTZ`.

Declaration

```
// C#  
public int Month{get;}
```

Property Value

A number that represents a month. Range of Month is (1 to 12).

Exceptions

`OracleNullValueException` - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

Nanosecond

This property gets the nanosecond component of an `OracleTimeStampLTZ`.

Declaration

```
// C#  
public int Nanosecond{get;}
```

Property Value

A number that represents a nanosecond. Range of `Nanosecond` is (0 to 999999999).

Exceptions

`OracleNullValueException` - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

Second

This property gets the second component of an `OracleTimeStampLTZ`.

Declaration

```
// C#  
public int Second{get;}
```

Property Value

A number that represents a second. Range of `Second` is (0 to 59).

Exceptions

`OracleNullValueException` - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

Value

This property specifies the date and time that is stored in the OracleTimeStampLTZ structure.

Declaration

```
// C#  
public DateTime Value{get;}
```

Property Value

A DateTime.

Exceptions

OracleNullValueException - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

Year

This property gets the year component of an OracleTimeStampLTZ.

Declaration

```
// C#  
public int Year{get;}
```

Property Value

A number that represents a year. The range of Year is (-4712 to 9999).

Exceptions

OracleNullValueException - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

OracleTimeStampLTZ Methods

The OracleTimeStampLTZ methods are listed in [Table 14-126](#).

Table 14-126 OracleTimeStampLTZ Methods

Methods	Description
AddDays	Adds the supplied number of days to the current instance
AddHours	Adds the supplied number of hours to the current instance
AddMilliseconds	Adds the supplied number of milliseconds to the current instance
AddMinutes	Adds the supplied number of minutes to the current instance
AddMonths	Adds the supplied number of months to the current instance
AddNanoseconds	Adds the supplied number of nanoseconds to the current instance
AddSeconds	Adds the supplied number of seconds to the current instance
AddYears	Adds the supplied number of years to the current instance
CompareTo	Compares the current OracleTimeStampLTZ instance to an object and returns an integer that represents their relative values
Equals	Determines whether or not an object has the same date and time as the current OracleTimeStampLTZ instance (Overloaded)
GetHashCode	Returns a hash code for the OracleTimeStampLTZ instance
GetDaysBetween	Subtracts an OracleTimeStampLTZ from the current instance and returns an OracleIntervalDS that represents the difference
GetYearsBetween	Subtracts an OracleTimeStampLTZ from the current instance and returns an OracleIntervalYM that represents the difference
GetType	Inherited from System.Object
ToOracleDate	Converts the current OracleTimeStampLTZ structure to an OracleDate structure
ToOracleTimeStamp	Converts the current OracleTimeStampLTZ structure to an OracleTimeStamp structure
ToOracleTimeStampTZ	Converts the current OracleTimeStampLTZ structure to an OracleTimeStampTZ structure
ToString	Converts the current OracleTimeStampLTZ structure to a string
ToUniversalTime	Converts the current local time to Coordinated Universal Time (UTC)

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

AddDays

This method adds the supplied number of days to the current instance.

Declaration

```
// C#  
public OracleTimeStampLTZ AddDays(double days);
```

Parameters

- *days*
The supplied number of days. Range is $(-1,000,000,000 < days < 1,000,000,000)$

Return Value

An OracleTimeStampLTZ.

Exceptions

OracleNullValueException - The current instance has a null value.

ArgumentOutOfRangeException - The argument value is out of the specified range.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

AddHours

This method adds the supplied number of hours to the current instance.

Declaration

```
// C#  
public OracleTimeStampLTZ AddHours(double hours);
```

Parameters

- *hours*
The supplied number of hours. Range is $(-24,000,000,000 < hours < 24,000,000,000)$.

Return Value

An OracleTimeStampLTZ.

Exceptions

OracleNullValueException - The current instance has a null value.

ArgumentOutOfRangeException - The argument value is out of the specified range.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

AddMilliseconds

This method adds the supplied number of milliseconds to the current instance.

Declaration

```
// C#  
public OracleTimeStampLTZ AddMilliseconds(double milliseconds);
```

Parameters

- *milliseconds*

The supplied number of milliseconds. Range is $(-8.64 * 10^{16} < \text{milliseconds} < 8.64 * 10^{16})$.

Return Value

An OracleTimeStampLTZ.

Exceptions

OracleNullValueException - The current instance has a null value.

ArgumentOutOfRangeException - The argument value is out of the specified range.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

AddMinutes

This method adds the supplied number of minutes to the current instance.

Declaration

```
// C#  
public OracleTimeStampLTZ AddMinutes(double minutes);
```

Parameters

- *minutes*
The supplied number of minutes. Range is $(-1,440,000,000,000 < minutes < 1,440,000,000,000)$.

Return Value

An OracleTimeStampLTZ.

Exceptions

OracleNullValueException - The current instance has a null value.

ArgumentOutOfRangeException - The argument value is out of the specified range.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

AddMonths

This method adds the supplied number of months to the current instance.

Declaration

```
// C#  
public OracleTimeStampLTZ AddMonths(long months);
```

Parameters

- *months*
The supplied number of months. Range is $(-12,000,000,000 < months < 12,000,000,000)$.

Return Value

An OracleTimeStampLTZ.

Exceptions

OracleNullValueException - The current instance has a null value.

ArgumentOutOfRangeException - The argument value is out of the specified range.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

AddNanoseconds

This method adds the supplied number of nanoseconds to the current instance.

Declaration

```
// C#  
public OracleTimeStampLTZ AddNanoseconds(long nanoseconds);
```

Parameters

- *nanoseconds*
The supplied number of nanoseconds.

Return Value

An OracleTimeStampLTZ.

Exceptions

OracleNullValueException - The current instance has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

AddSeconds

This method adds the supplied number of seconds to the current instance.

Declaration

```
// C#  
public OracleTimeStampLTZ AddSeconds(double seconds);
```

Parameters

- *seconds*
The supplied number of seconds. Range is $(-8.64 * 10^{13} < \text{seconds} < 8.64 * 10^{13})$.

Return Value

An OracleTimeStampLTZ.

Exceptions

OracleNullValueException - The current instance has a null value.

ArgumentOutOfRangeException - The argument value is out of the specified range.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

AddYears

This method adds the supplied number of years to the current instance

Declaration

```
// C#  
public OracleTimeStampLTZ AddYears(int years);
```

Parameters

- *years*
The supplied number of years. Range is (-999,999,999 <= *years* <= 999,999,999)

Return Value

An OracleTimeStampLTZ.

Exceptions

OracleNullValueException - The current instance has a null value.

ArgumentOutOfRangeException - The argument value is out of the specified range.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

CompareTo

This method compares the current `OracleTimeStampLTZ` instance to an object, and returns an integer that represents their relative values.

Declaration

```
// C#  
public int CompareTo(object obj);
```

Parameters

- *obj*
The object being compared to the current `OracleTimeStampLTZ` instance.

Return Value

The method returns a number that is:

- Less than zero: if the current `OracleTimeStampLTZ` instance value is less than that of *obj*.
- Zero: if the current `OracleTimeStampLTZ` instance and *obj* values are equal.
- Greater than zero: if the current `OracleTimeStampLTZ` instance value is greater than that of *obj*.

Implements

`Comparable`

Exceptions

`ArgumentException` - The *obj* parameter is not of type `OracleTimeStampLTZ`.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between `OracleTimeStampLTZ`s. For example, comparing an `OracleTimeStampLTZ` instance with an `OracleBinary` instance is not allowed. When an `OracleTimeStampLTZ` is compared with a different type, an `ArgumentException` is thrown.
- Any `OracleTimeStampLTZ` that has a value is greater than an `OracleTimeStampLTZ` that has a null value.
- Two `OracleTimeStampLTZ`s that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

Equals

Overrides Object

This method determines whether or not an object has the same date and time as the current OracleTimeStampLTZ instance.

Declaration

```
// C#  
public override bool Equals(object obj);
```

Parameters

- *obj*

The object being compared to the current OracleTimeStampLTZ instance.

Return Value

Returns true if the *obj* is of type OracleTimeStampLTZ and represents the same date and time; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleTimeStampLTZ that has a value is greater than an OracleTimeStampLTZ that has a null value.
- Two OracleTimeStampLTZs that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

GetHashCode

Overrides Object

This method returns a hash code for the OracleTimeStampLTZ instance.

Declaration

```
// C#  
public override int GetHashCode();
```

Return Value

A number that represents the hash code.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

GetDaysBetween

This method subtracts an `OracleTimeStampLTZ` value from the current instance and returns an `OracleIntervalDS` that represents the difference.

Declaration

```
// C#  
public OracleIntervalDS GetDaysBetween(OracleTimeStampLTZ value1);
```

Parameters

- *value1*
The `OracleTimeStampLTZ` value being subtracted.

Return Value

An `OracleIntervalDS` that represents the interval between two `OracleTimeStampLTZ` values.

Remarks

If either the current instance or the parameter has a null value, the returned `OracleIntervalDS` has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

GetYearsBetween

This method subtracts an `OracleTimeStampLTZ` value from the current instance and returns an `OracleIntervalYM` that represents the time interval.

Declaration

```
// C#  
public OracleIntervalYM GetYearsBetween(OracleTimeStampLTZ value1);
```

Parameters

- *value1*

The OracleTimeStampLTZ value being subtracted.

Return Value

An OracleIntervalYM that represents the interval between two OracleTimeStampLTZ values.

Remarks

If either the current instance or the parameter has a null value, the returned OracleIntervalYM has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

ToOracleDate

This method converts the current OracleTimeStampLTZ structure to an OracleDate structure.

Declaration

```
// C#  
public OracleDate ToOracleDate();
```

Return Value

The returned OracleDate structure contains the date and time in the current instance.

Remarks

The precision of the OracleTimeStampLTZ value can be lost during the conversion.

If the current instance has a null value, the value of the returned OracleDate structure has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

ToOracleTimeStamp

This method converts the current OracleTimeStampLTZ structure to an OracleTimeStamp structure.

Declaration

```
// C#  
public OracleTimeStamp ToOracleTimeStamp();
```

Return Value

The returned `OracleTimeStamp` contains the date and time in the current instance.

Remarks

If the current instance has a null value, the value of the returned `OracleTimeStamp` structure has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

ToOracleTimeStampTZ

This method converts the current `OracleTimeStampLTZ` structure to an `OracleTimeStampTZ` structure.

Declaration

```
// C#  
public OracleTimeStampTZ ToOracleTimeStampTZ();
```

Return Value

The returned `OracleTimeStampTZ` contains the date and time of the current instance, with the time zone set to the `OracleGlobalization.TimeZone` from the thread.

Remarks

If the current instance has a null value, the value of the returned `OracleTimeStampTZ` structure has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)

ToString

Overrides Object

This method converts the current OracleTimeStampLTZ structure to a string.

Declaration

```
// C#  
public override string ToString();
```

Return Value

A string that represents the same date and time as the current OracleTimeStampLTZ structure.

Remarks

The returned value is a string representation of the OracleTimeStampLTZ in the format specified by the OracleGlobalization.TimeStampFormat property of the thread.

The names and abbreviations used for months and days are in the language specified by the DateLanguage and Calendar properties of the thread's OracleGlobalization object. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Types;  
using Oracle.DataAccess.Client;  
  
class ToStringSample  
{  
    static void Main()  
    {  
        // Set the nls_timestamp_format for the OracleTimeStampLTZ(string)  
        // constructor  
        OracleGlobalization info = OracleGlobalization.GetClientInfo();  
        info.TimeStampFormat = "DD-MON-YYYY HH:MI:SS.FF AM";  
        OracleGlobalization.SetThreadInfo(info);  
  
        // construct OracleTimeStampLTZ from a string using the format  
        // specified.  
        OracleTimeStampLTZ ts =  
            new OracleTimeStampLTZ("11-NOV-1999 11:02:33.444 AM");  
  
        // Set the nls_timestamp_format for the ToString() method  
        info.TimeStampFormat = "YYYY-MON-DD HH:MI:SS.FF AM";  
        OracleGlobalization.SetThreadInfo(info);  
  
        // Prints "1999-NOV-11 11:02:33.444000000 AM"  
        Console.WriteLine(ts.ToString());  
    }  
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)

ToUniversalTime

This method converts the current local time to Coordinated Universal Time (UTC).

Declaration

```
// C#  
public OracleTimeStampTZ ToUniversalTime();
```

Return Value

An `OracleTimeStampTZ` structure.

Remarks

If the current instance has a null value, the value of the returned `OracleTimeStampTZ` structure has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampLTZ Structure](#)
- [OracleTimeStampLTZ Members](#)

OracleTimeStampTZ Structure

The `OracleTimeStampTZ` structure represents the Oracle `TIMESTAMP WITH TIME ZONE` data type to be stored in or retrieved from a database. Each `OracleTimeStampTZ` stores the following information: year, month, day, hour, minute, second, nanosecond, and time zone.

Class Inheritance

`System.Object`

`System.ValueType`

`Oracle.DataAccess.Types.OracleTimeStampTZ`

Declaration

```
// C#
public struct OracleTimeStampTZ : IComparable, INullable, IXmlSerializable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class OracleTimeStampTZSample
{
    static void Main()
    {
        // Set the nls parameters for the current thread
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.TimeZone = "US/Eastern";
        info.TimestampFormat = "DD-MON-YYYY HH:MI:SS.FF AM";
        info.TimestampTZFormat = "DD-MON-YYYY HH:MI:SS.FF AM TZR";
        OracleGlobalization.SetThreadInfo(info);

        // Create an OracleTimeStampTZ in US/Pacific time zone
        OracleTimeStampTZ tstz1=new OracleTimeStampTZ("11-NOV-1999 "+"
            "11:02:33.444 AM US/Pacific");

        // Note that ToOracleTimeStampTZ uses the thread's time zone region,
        // "US/Eastern"
        OracleTimeStamp ts = new OracleTimeStamp("11-NOV-1999 11:02:33.444 AM");
        OracleTimeStampTZ tstz2 = ts.ToOracleTimeStampTZ();

        // Calculate the difference between tstz1 and tstz2
        OracleIntervalDS idsDiff = tstz1.GetDaysBetween(tstz2);

        // Display information
        Console.WriteLine("tstz1.TimeZone = " + tstz1.TimeZone);

        // Prints "US/Pacific"
```

```
Console.WriteLine("tstz2.TimeZone = " + tstz2.TimeZone);

// Prints "US/Eastern"
Console.WriteLine("idsDiff.Hours = " + idsDiff.Hours); // Prints 3
Console.WriteLine("idsDiff.Minutes = " + idsDiff.Minutes); // Prints 0
}
}
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Members](#)
- [OracleTimeStampTZ Constructors](#)
- [OracleTimeStampTZ Static Fields](#)
- [OracleTimeStampTZ Static Methods](#)
- [OracleTimeStampTZ Static Operators](#)
- [OracleTimeStampTZ Static Type Conversions](#)
- [OracleTimeStampTZ Properties](#)
- [OracleTimeStampTZ Methods](#)

OracleTimeStampTZ Members

OracleTimeStampTZ members are listed in the following tables:

OracleTimeStampTZ Constructors

OracleTimeStampTZ constructors are listed in [Table 14-127](#)

Table 14-127 OracleTimeStampTZ Constructors

Constructor	Description
OracleTimeStampTZ Constructors	Instantiates a new instance of OracleTimeStampTZ structure (Overloaded)

OracleTimeStampTZ Static Fields

The OracleTimeStampTZ static fields are listed in [Table 14-128](#).

Table 14-128 OracleTimeStampTZ Static Fields

Field	Description
MaxValue	Represents the maximum valid date for an OracleTimeStampTZ structure in UTC, which is December 31, 999923:59:59.999999999
MinValue	Represents the minimum valid date for an OracleTimeStampTZ structure in UTC, which is January 1, -4712 0:0:0

Table 14-128 (Cont.) OracleTimeStampTZ Static Fields

Field	Description
Null	Represents a null value that can be assigned to an instance of the OracleTimeStampTZ structure

OracleTimeStampTZ Static Methods

The OracleTimeStampTZ static methods are listed in [Table 14-129](#).

Table 14-129 OracleTimeStampTZ Static Methods

Methods	Description
Equals	Determines if two OracleTimeStampTZ values are equal (Overloaded)
GetSysDate	Gets an OracleTimeStampTZ structure that represents the current date and time
GreaterThan	Determines if the first of two OracleTimeStampTZ values is greater than the second
GreaterThanOrEqual	Determines if the first of two OracleTimeStampTZ values is greater than or equal to the second
LessThan	Determines if the first of two OracleTimeStampTZ values is less than the second
LessThanOrEqual	Determines if the first of two OracleTimeStampTZ values is less than or equal to the second
NotEquals	Determines if two OracleTimeStampTZ values are not equal
Parse	Gets an OracleTimeStampTZ structure and sets its value for date and time using the supplied string
SetPrecision	Returns a new instance of an OracleTimeStampTZ with the specified fractional second precision

OracleTimeStampTZ Static Operators

The OracleTimeStampTZ static operators are listed in [Table 14-130](#).

Table 14-130 OracleTimeStampTZ Static Operators

Operator	Description
operator +	Adds the supplied instance value to the supplied OracleTimeStampTZ and returns a new OracleTimeStampTZ structure (Overloaded)
operator ==	Determines if two OracleTimeStampTZ values are equal
operator >	Determines if the first of two OracleTimeStampTZ values is greater than the second
operator >=	Determines if the first of two OracleTimeStampTZ values is greater than or equal to the second

Table 14-130 (Cont.) OracleTimeStampTZ Static Operators

Operator	Description
operator !=	Determines if two OracleTimeStampTZ values are not equal
operator <	Determines if the first of two OracleTimeStampTZ values is less than the second
operator <=	Determines if the first of two OracleTimeStampTZ values is less than or equal to the second
operator -	Subtracts the supplied instance value from the supplied OracleTimeStampTZ and returns a new OracleTimeStampTZ structure (Overloaded)

OracleTimeStampTZ Static Type Conversions

The OracleTimeStampTZ static type conversions are listed in [Table 14-131](#).

Table 14-131 OracleTimeStampTZ Static Type Conversions

Operator	Description
explicit operator OracleTimeStampTZ	Converts an instance value to an OracleTimeStampTZ structure (Overloaded)
implicit operator OracleTimeStampTZ	Converts an instance value to an OracleTimeStampTZ structure (Overloaded)
explicit operator DateTime	Converts an OracleTimeStampTZ value to a DateTime structure

OracleTimeStampTZ Properties

The OracleTimeStampTZ properties are listed in [Table 14-132](#).

Table 14-132 OracleTimeStampTZ Properties

Properties	Description
BinData	Returns an array of bytes that represents an Oracle TIMESTAMP WITH TIME ZONE in Oracle internal format
Day	Specifies the day component of an OracleTimeStampTZ in the current time zone
IsNull	Indicates whether or not the current instance has a null value
Hour	Specifies the hour component of an OracleTimeStampTZ in the current time zone
Millisecond	Specifies the millisecond component of an OracleTimeStampTZ in the current time zone
Minute	Specifies the minute component of an OracleTimeStampTZ in the current time zone
Month	Specifies the month component of an OracleTimeStampTZ in the current time zone

Table 14-132 (Cont.) OracleTimeStampTZ Properties

Properties	Description
Nanosecond	Specifies the nanosecond component of an OracleTimeStampTZ in the current time zone
Second	Specifies the second component of an OracleTimeStampTZ in the current time zone
TimeZone	Returns the time zone of the OracleTimeStampTZ instance
Value	Returns the date and time that is stored in the OracleTimeStampTZ structure in the current time zone
Year	Specifies the year component of an OracleTimeStampTZ

OracleTimeStampTZ Methods

The OracleTimeStampTZ methods are listed in [Table 14-133](#).

Table 14-133 OracleTimeStampTZ Methods

Methods	Description
AddDays	Adds the supplied number of days to the current instance
AddHours	Adds the supplied number of hours to the current instance
AddMilliseconds	Adds the supplied number of milliseconds to the current instance
AddMinutes	Adds the supplied number of minutes to the current instance
AddMonths	Adds the supplied number of months to the current instance
AddNanoseconds	Adds the supplied number of nanoseconds to the current instance
AddSeconds	Adds the supplied number of seconds to the current instance
AddYears	Adds the supplied number of years to the current instance
CompareTo	Compares the current OracleTimeStampTZ instance to an object, and returns an integer that represents their relative values
Equals	Determines whether or not an object has the same date and time as the current OracleTimeStampTZ instance
GetDaysBetween	Subtracts an OracleTimeStampTZ from the current instance and returns an OracleIntervalDS that represents the time interval
GetHashCode	Returns a hash code for the OracleTimeStampTZ instance
GetTimeZoneOffset	Gets the time zone information in hours and minutes of the current OracleTimeStampTZ

Table 14-133 (Cont.) OracleTimeStampTZ Methods

Methods	Description
GetYearsBetween	Subtracts an OracleTimeStampTZ from the current instance and returns an OracleIntervalYM that represents the time interval
GetType	Inherited from System.Object
ToLocalTime	Converts the current OracleTimeStampTZ instance to local time
ToOracleDate	Converts the current OracleTimeStampTZ structure to an OracleDate structure
ToOracleTimeStampLTZ	Converts the current OracleTimeStampTZ structure to an OracleTimeStampLTZ structure
ToOracleTimeStamp	Converts the current OracleTimeStampTZ structure to an OracleTimeStamp structure
ToString	Converts the current OracleTimeStampTZ structure to a string
ToUniversalTime	Converts the current datetime to Coordinated Universal Time (UTC)

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)

OracleTimeStampTZ Constructors

The OracleTimeStampTZ constructors create new instances of the OracleTimeStampTZ structure.

Overload List:

- [OracleTimeStampTZ\(DateTime\)](#)
This constructor creates a new instance of the OracleTimeStampTZ structure and sets its value for date and time using the supplied DateTime value.
- [OracleTimeStampTZ\(DateTime, string\)](#)
This constructor creates a new instance of the OracleTimeStampTZ structure and sets its value for date and time using the supplied DateTime value and the supplied time zone data.
- [OracleTimeStampTZ\(string\)](#)
This constructor creates a new instance of the OracleTimeStampTZ structure and sets its value for date and time using the supplied string.
- [OracleTimeStampTZ\(int, int, int\)](#)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, and day.

- [OracleTimeStampTZ\(int, int, int, string\)](#)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, and time zone data.

- [OracleTimeStampTZ\(int, int, int, int, int, int\)](#)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, and second.

- [OracleTimeStampTZ\(int, int, int, int, int, int, string\)](#)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, and time zone data.

- [OracleTimeStampTZ\(int, int, int, int, int, int, double\)](#)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, and millisecond.

- [OracleTimeStampTZ\(int, int, int, int, int, int, double, string\)](#)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, millisecond, and time zone data.

- [OracleTimeStampTZ\(int, int, int, int, int, int, int\)](#)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, and nanosecond.

- [OracleTimeStampTZ\(int, int, int, int, int, int, int, string\)](#)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, nanosecond, and time zone data.

- [OracleTimeStampTZ\(byte \[\]\)](#)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value to the provided byte array, that represents the internal Oracle `TIMESTAMP WITH TIME ZONE` format.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

OracleTimeStampTZ(DateTime)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using the supplied `DateTime` value.

Declaration

```
// C#  
public OracleTimeStampTZ (DateTime dt);
```

Parameters

- *dt*
The supplied DateTime value.

Remarks

The time zone is set to the `OracleGlobalization.TimeZone` of the thread.

Exceptions

`ArgumentException` - The *dt* parameter cannot be used to construct a valid `OracleTimeStampTZ`.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

OracleTimeStampTZ(DateTime, string)

This constructor creates a new instance of the `OracleTimeStampTZ` structure with the supplied `DateTime` value and the time zone data.

Declaration

```
// C#  
public OracleTimeStampTZ (DateTime value1, string timeZone);
```

Parameters

- *value1*
The supplied DateTime value.
- *timeZone*
The time zone data provided.

Exceptions

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampTZ`.

Remarks

timeZone can be either an hour offset, for example, 7:00, or a valid time zone region name that is provided in `V$TIMEZONE_NAMES`, such as `US/Pacific`. Time zone abbreviations are not supported.

If time zone is null, the `OracleGlobalization.TimeZone` of the thread is used.

Note

PST is a time zone region name as well as a time zone abbreviation; therefore it is accepted by `OracleTimeStampTZ`.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

OracleTimeStampTZ(string)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using the supplied string.

Declaration

```
// C#  
public OracleTimeStampTZ (string tsStr);
```

Parameters

- *tsStr*
A string that represents an Oracle `TIMESTAMP WITH TIME ZONE`.

Exceptions

`ArgumentException` - The *tsStr* is an invalid string representation of an Oracle `TIMESTAMP WITH TIME ZONE` or the *tsStr* is not in the timestamp format specified by the `OracleGlobalization.TimestampTZFormat` property of the thread.

`ArgumentNullException` - The *tsStr* value is null.

Remarks

The names and abbreviations used for months and days are in the language specified by the `DateLanguage` and `Calendar` properties of the thread's `OracleGlobalization` object. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Client;  
using Oracle.DataAccess.Types;  
  
class OracleTimeStampTZSample
```

```
{
    static void Main()
    {
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.TimeStampTZFormat = "DD-MON-YYYY HH:MI:SS.FF AM TZR";
        OracleGlobalization.SetThreadInfo(info);

        // construct OracleTimeStampTZ from a string using the format specified.
        OracleTimeStampTZ tstz = new OracleTimeStampTZ("11-NOV-1999" +
            "11:02:33.444 AM US/Pacific");

        // Set the nls_timestamp_tz_format for the ToString() method
        info.TimeStampTZFormat = "YYYY-MON-DD HH:MI:SS.FF AM TZR";
        OracleGlobalization.SetThreadInfo(info);

        // Prints "1999-NOV-11 11:02:33.444000000 AM US/Pacific"
        Console.WriteLine(tstz.ToString());
    }
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)
- *Oracle AI Database SQL Language Reference* for further information on date format elements

OracleTimeStampTZ(int, int, int)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, and day.

Declaration

```
// C#
public OracleTimeStampTZ(int year, int month, int day);
```

Parameters

- *year*
The year provided. Range of *year* is (-4712 to 9999).
- *month*
The month provided. Range of *month* is (1 to 12).
- *day*
The day provided. Range of *day* is (1 to 31).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampTZ` (that is, the day is out of range for the month).

Remarks

The time zone is set to the `OracleGlobalization.TimeZone` of the thread.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

OracleTimeStampTZ(int, int, int, string)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, and time zone data.

Declaration

```
// C#
public OracleTimeStampTZ(int year, int month, int day,
    string timeZone);
```

Parameters

- *year*
The year provided. Range of *year* is (-4712 to 9999).
- *month*
The month provided. Range of *month* is (1 to 12).
- *day*
The day provided. Range of *day* is (1 to 31).
- *timeZone*
The time zone data provided.

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampTZ` (that is, the day is out of range for the month or the time zone is invalid).

Remarks

timeZone can be either an hour offset, for example, 7:00, or a valid time zone region name that is provided in V\$TIMEZONE_NAMES, such as US/Pacific. Time zone abbreviations are not supported.

If time zone is null, the OracleGlobalization.TimeZone of the thread is used.

Note

PST is a time zone region name as well as a time zone abbreviation; therefore it is accepted by OracleTimeStampTZ.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

OracleTimeStampTZ(int, int, int, int, int, int)

This constructor creates a new instance of the OracleTimeStampTZ structure and sets its value for date and time using year, month, day, hour, minute, and second.

Declaration

```
// C#  
public OracleTimeStampTZ(int year, int month, int day, int hour,  
    int minute, int second);
```

Parameters

- *year*
The year provided. Range of *year* is (-4712 to 9999).
- *month*
The month provided. Range of *month* is (1 to 12).
- *day*
The day provided. Range of *day* is (1 to 31).
- *hour*
The hour provided. Range of *hour* is (0 to 23).
- *minute*
The minute provided. Range of *minute* is (0 to 59).
- *second*
The second provided. Range of *second* is (0 to 59).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampTZ` (that is, the day is out of range for the month).

Remarks

The time zone is set to the `OracleGlobalization.TimeZone` of the thread.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

OracleTimeStampTZ(int, int, int, int, int, int, string)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, and time zone data.

Declaration

```
// C#
public OracleTimeStampTZ (int year, int month, int day, int hour,
    int minute, int second, string timeZone);
```

Parameters

- *year*
The year provided. Range of *year* is (-4712 to 9999).
- *month*
The month provided. Range of *month* is (1 to 12).
- *day*
The day provided. Range of *day* is (1 to 31).
- *hour*
The hour provided. Range of *hour* is (0 to 23).
- *minute*
The minute provided. Range of *minute* is (0 to 59).
- *second*
The second provided. Range of *second* is (0 to 59).
- *timeZone*
The time zone data provided.

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampTZ` (that is, the day is out of range of the month or the time zone is invalid).

Remarks

`timeZone` can be either an hour offset, for example, 7:00, or a valid time zone region name that is provided in `V$TIMEZONE_NAMES`, such as `US/Pacific`. Time zone abbreviations are not supported.

If time zone is null, the `OracleGlobalization.TimeZone` of the thread is used.

Note

PST is a time zone region name as well as a time zone abbreviation; therefore it is accepted by `OracleTimeStampTZ`.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

OracleTimeStampTZ(int, int, int, int, int, double)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, and millisecond.

Declaration

```
// C#  
public OracleTimeStampTZ(int year, int month, int day, int hour,  
    int minute, int second, double millisecond);
```

Parameters

- *year*
The year provided. Range of *year* is (-4712 to 9999).
- *month*
The month provided. Range of *month* is (1 to 12).
- *day*
The day provided. Range of *day* is (1 to 31).
- *hour*

The hour provided. Range of *hour* is (0 to 23).

- *minute*

The minute provided. Range of *minute* is (0 to 59).

- *second*

The second provided. Range of *second* is (0 to 59).

- *millisecond*

The millisecond provided. Range of *millisecond* is (0 to 999.999999).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampTZ` (that is, the day is out of range for the month).

Remarks

The time zone is set to the `OracleGlobalization.TimeZone` of the thread.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

OracleTimeStampTZ(int, int, int, int, int, int, double, string)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, millisecond, and time zone data.

Declaration

```
// C#
public OracleTimeStampTZ(int year, int month, int day, int hour,
    int minute, int second, double millisecond, string timeZone);
```

Parameters

- *year*

The year provided. Range of *year* is (-4712 to 9999).

- *month*

The month provided. Range of *month* is (1 to 12).

- *day*

The day provided. Range of *day* is (1 to 31).

- *hour*

The hour provided. Range of *hour* is (0 to 23).

- *minute*

The minute provided. Range of *minute* is (0 to 59).

- *second*

The second provided. Range of *second* is (0 to 59).

- *millisecond*

The millisecond provided. Range of *millisecond* is (0 to 999.999999).

- *timeZone*

The time zone data provided.

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampTZ` (that is, the day is out of range for the month or the time zone is invalid).

Remarks

timeZone can be either an hour offset, for example, 7:00, or a valid time zone region name that is provided in `V$TIMEZONE_NAMES`, such as `US/Pacific`. Time zone abbreviations are not supported.

If time zone is null, the `OracleGlobalization.TimeZone` of the thread is used.

Note

PST is a time zone region name as well as a time zone abbreviation; therefore it is accepted by `OracleTimeStampTZ`.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

OracleTimeStampTZ(int, int, int, int, int, int, int)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, and nanosecond.

Declaration

```
// C#  
public OracleTimeStampTZ(int year, int month, int day, int hour,  
    int minute, int second, int nanosecond);
```

Parameters

- *year*
The year provided. Range of *year* is (-4712 to 9999).
- *month*
The month provided. Range of *month* is (1 to 12).
- *day*
The day provided. Range of *day* is (1 to 31).
- *hour*
The hour provided. Range of *hour* is (0 to 23).
- *minute*
The minute provided. Range of *minute* is (0 to 59).
- *second*
The second provided. Range of *second* is (0 to 59).
- *nanosecond*
The nanosecond provided. Range of *nanosecond* is (0 to 999999999).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampTZ` (that is, the day is out of range for the month).

Remarks

The time zone is set to the `OracleGlobalization.TimeZone` of the thread.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

OracleTimeStampTZ(int, int, int, int, int, int, string)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, nanosecond, and time zone data.

Declaration

```
// C#  
public OracleTimeStampTZ(int year, int month, int day, int hour,  
    int minute, int second, int nanosecond, string timeZone);
```

Parameters

- *year*
The year provided. Range of *year* is (-4712 to 9999).
- *month*
The month provided. Range of *month* is (1 to 12).
- *day*
The day provided. Range of *day* is (1 to 31).
- *hour*
The hour provided. Range of *hour* is (0 to 23).
- *minute*
The minute provided. Range of *minute* is (0 to 59).
- *second*
The second provided. Range of *second* is (0 to 59).
- *nanosecond*
The nanosecond provided. Range of *nanosecond* is (0 to 999999999).
- *timeZone*
The time zone data provided.

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampTZ` (that is, the day is out of range for the month or the time zone is invalid).

Remarks

timeZone can be either an hour offset, for example, 7:00, or a valid time zone region name that is provided in `V$TIMEZONE_NAMES`, such as `US/Pacific`. Time zone abbreviations are not supported.

If time zone is null, the `OracleGlobalization.TimeZone` of the thread is used.

Note

PST is a time zone region name as well as a time zone abbreviation; therefore it is accepted by `OracleTimeStampTZ`.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

OracleTimeStampTZ(byte [])

This constructor creates a new instance of the OracleTimeStampTZ structure and sets its value to the provided byte array, that represents the internal Oracle TIMESTAMP WITH TIME ZONE format.

Declaration

```
// C#
public OracleTimeStampLTZ (byte[] bytes);
```

Parameters

- *bytes*
The provided byte array that represents an Oracle TIMESTAMP WITH TIME ZONE in Oracle internal format.

Exceptions

ArgumentException - *bytes* is not in internal Oracle TIMESTAMP WITH TIME ZONE format or *bytes* is not a valid Oracle TIMESTAMP WITH TIME ZONE.
ArgumentNullException - *bytes* is null.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

OracleTimeStampTZ Static Fields

The OracleTimeStampTZ static fields are listed in [Table 14-134](#).

Table 14-134 OracleTimeStampTZ Static Fields

Field	Description
MaxValue	Represents the maximum valid date for an OracleTimeStampTZ structure in UTC, which is December 31, 999923:59:59.999999999
MinValue	Represents the minimum valid date for an OracleTimeStampTZ structure in UTC, which is January 1, -4712 0:0:0

Table 14-134 (Cont.) OracleTimeStampTZ Static Fields

Field	Description
Null	Represents a null value that can be assigned to an instance of the OracleTimeStampTZ structure

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

MaxValue

This static field represents the maximum valid datetime time for an OracleTimeStampTZ structure in UTC, which is December 31, 999923:59:59.999999999.

Declaration

```
// C#
public static readonly OracleTimeStampTZ MaxValue;
```

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

MinValue

This static field represents the minimum valid datetime for an OracleTimeStampTZ structure in UTC, which is January 1, -4712 0:0:0.

Declaration

```
// C#
public static readonly OracleTimeStampTZ MinValue;
```

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

Null

This static field represents a null value that can be assigned to an instance of the `OracleTimeStampTZ` structure.

Declaration

```
// C#  
public static readonly OracleTimeStampTZ Null;
```

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

OracleTimeStampTZ Static Methods

The `OracleTimeStampTZ` static methods are listed in [Table 14-135](#).

Table 14-135 OracleTimeStampTZ Static Methods

Methods	Description
Equals	Determines if two <code>OracleTimeStampTZ</code> values are equal (Overloaded)
GetSysDate	Gets an <code>OracleTimeStampTZ</code> structure that represents the current date and time
GreaterThan	Determines if the first of two <code>OracleTimeStampTZ</code> values is greater than the second
GreaterThanOrEqual	Determines if the first of two <code>OracleTimeStampTZ</code> values is greater than or equal to the second
LessThan	Determines if the first of two <code>OracleTimeStampTZ</code> values is less than the second
LessThanOrEqual	Determines if the first of two <code>OracleTimeStampTZ</code> values is less than or equal to the second
NotEquals	Determines if two <code>OracleTimeStampTZ</code> values are not equal
Parse	Gets an <code>OracleTimeStampTZ</code> structure and sets its value for date and time using the supplied string
SetPrecision	Returns a new instance of an <code>OracleTimeStampTZ</code> with the specified fractional second precision

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

Equals

This static method determines if two `OracleTimeStampTZ` values are equal.

Declaration

```
// C#  
public static bool Equals(OracleTimeStampTZ value1,  
    OracleTimeStampTZ value2);
```

Parameters

- *value1*
The first `OracleTimeStampTZ`.
- *value2*
The second `OracleTimeStampTZ`.

Return Value

Returns true if two `OracleTimeStampTZ` values are equal. Returns false otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.
- Two `OracleTimeStampTZ`s that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

GetSysDate

This static method gets an `OracleTimeStampTZ` structure that represents the current date and time.

Declaration

```
// C#  
public static OracleTimeStampTZ GetSysDate();
```

Return Value

An `OracleTimeStampTZ` structure that represents the current date and time.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

GreaterThan

This static method determines if the first of two `OracleTimeStampTZ` values is greater than the second.

Declaration

```
// C#  
public static bool GreaterThan(OracleTimeStampTZ value1,  
    OracleTimeStampTZ value2);
```

Parameters

- *value1*
The first `OracleTimeStampTZ`.
- *value2*
The second `OracleTimeStampTZ`.

Return Value

Returns `true` if the first of two `OracleTimeStampTZ` values is greater than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.
- Two `OracleTimeStampTZ`s that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

GreaterThanOrEqual

This static method determines if the first of two `OracleTimeStampTZ` values is greater than or equal to the second.

Declaration

```
// C#  
public static bool GreaterThanOrEqual(OracleTimeStampTZ value1,  
    OracleTimeStampTZ value2);
```

Parameters

- *value1*
The first `OracleTimeStampTZ`.
- *value2*
The second `OracleTimeStampTZ`.

Return Value

Returns `true` if the first of two `OracleTimeStampTZ` values is greater than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.
- Two `OracleTimeStampTZ`s that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

LessThan

This static method determines if the first of two `OracleTimeStampTZ` values is less than the second.

Declaration

```
// C#  
public static bool LessThan(OracleTimeStampTZ value1,  
    OracleTimeStampTZ value2);
```

Parameters

- *value1*
The first OracleTimeStampTZ.
- *value2*
The second OracleTimeStampTZ.

Return Value

Returns true if the first of two OracleTimeStampTZ values is less than the second. Returns false otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any OracleTimeStampTZ that has a value is greater than an OracleTimeStampTZ that has a null value.
- Two OracleTimeStampTZs that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

LessThanOrEqual

This static method determines if the first of two OracleTimeStampTZ values is less than or equal to the second.

Declaration

```
// C#  
public static bool LessThanOrEqual(OracleTimeStampTZ value1,  
    OracleTimeStampTZ value2);
```

Parameters

- *value1*
The first OracleTimeStampTZ.
- *value2*
The second OracleTimeStampTZ.

Return Value

Returns true if the first of two `OracleTimeStampTZ` values is less than or equal to the second. Returns false otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.
- Two `OracleTimeStampTZ`s that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

NotEquals

This static method determines if two `OracleTimeStampTZ` values are not equal.

Declaration

```
// C#  
public static bool NotEquals(OracleTimeStampTZ value1,  
    OracleTimeStampTZ value2);
```

Parameters

- *value1*
The first `OracleTimeStampTZ`.
- *value2*
The second `OracleTimeStampTZ`.

Return Value

Returns true if two `OracleTimeStampTZ` values are not equal. Returns false otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.
- Two `OracleTimeStampTZ`s that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

Parse

This static method returns an `OracleTimeStampTZ` structure and sets its value for date and time using the supplied string.

Declaration

```
// C#  
public static OracleTimeStampTZ Parse(string tsStr);
```

Parameters

- *tsStr*
A string that represents an Oracle TIMESTAMP WITH TIME ZONE.

Return Value

An `OracleTimeStampTZ` structure.

Exceptions

`ArgumentException` - The *tsStr* is an invalid string representation of an Oracle TIMESTAMP WITH TIME ZONE or the *tsStr* is not in the timestamp format specified by the `OracleGlobalization.TimeStampTZFormat` property of the thread, which represents the Oracle `NLS_TIMESTAMP_TZ_FORMAT` parameter.

`ArgumentNullException` - The *tsStr* value is null.

Remarks

The names and abbreviations used for months and days are in the language specified by the `DateLanguage` and `Calendar` properties of the thread's `OracleGlobalization` object. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Client;  
using Oracle.DataAccess.Types;  
  
class ParseSample  
{  
    static void Main()  
    {  
        // Set the nls_timestamp_tz_format for the Parse() method  
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
```

```
info.TimeStampTZFormat = "DD-MON-YYYY HH:MI:SS.FF AM TZR";
OracleGlobalization.SetThreadInfo(info);

// construct OracleTimeStampTZ from a string using the format specified.
OracleTimeStampTZ tstz = OracleTimeStampTZ.Parse("11-NOV-1999 " +
    "11:02:33.444 AM US/Pacific");

// Set the nls_timestamp_tz_format for the ToString() method
info.TimeStampTZFormat = "YYYY-MON-DD HH:MI:SS.FF AM TZR";
OracleGlobalization.SetThreadInfo(info);

// Prints "1999-NOV-11 11:02:33.444000000 AM US/Pacific"
Console.WriteLine(tstz.ToString());
    }
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support"](#)

SetPrecision

This static method returns a new instance of an `OracleTimeStampTZ` with the specified fractional second precision.

Declaration

```
// C#
public static OracleTimeStampTZ SetPrecision(OracleTimeStampTZ value1,
    int fracSecPrecision);
```

Parameters

- *value1*
The provided `OracleTimeStampTZ` object.
- *fracSecPrecision*
The fractional second precision provided. Range of fractional second precision is (0 to 9).

Return Value

An `OracleTimeStampTZ` structure with the specified fractional second precision

Exceptions

`ArgumentOutOfRangeException` - *fracSecPrecision* is out of the specified range.

Remarks

The value specified in the supplied *fracSecPrecision* is used to perform a rounding off operation on the supplied `OracleTimeStampTZ` value. Depending on this value, 0 or more trailing zeros are displayed in the string returned by `ToString()`.

Example

The `OracleTimeStampTZ` with a value of "December 31, 9999 23:59:59.99 US/Pacific" results in the string "December 31, 9999 23:59:59.99000 US/Pacific" when `SetPrecision()` is called with the fractional second precision set to 5.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

OracleTimeStampTZ Static Operators

The `OracleTimeStampTZ` static operators are listed in [Table 14-136](#).

Table 14-136 OracleTimeStampTZ Static Operators

Operator	Description
operator +	Adds the supplied instance value to the supplied <code>OracleTimeStampTZ</code> and returns a new <code>OracleTimeStampTZ</code> structure (Overloaded)
operator ==	Determines if two <code>OracleTimeStampTZ</code> values are equal
operator >	Determines if the first of two <code>OracleTimeStampTZ</code> values is greater than the second
operator >=	Determines if the first of two <code>OracleTimeStampTZ</code> values is greater than or equal to the second
operator !=	Determines if two <code>OracleTimeStampTZ</code> values are not equal
operator <	Determines if the first of two <code>OracleTimeStampTZ</code> values is less than the second
operator <=	Determines if the first of two <code>OracleTimeStampTZ</code> values is less than or equal to the second
operator -	Subtracts the supplied instance value from the supplied <code>OracleTimeStampTZ</code> and returns a new <code>OracleTimeStampTZ</code> structure (Overloaded)

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

operator +

operator+ adds the supplied structure to the supplied OracleTimeStampTZ and returns a new OracleTimeStampTZ structure.

Overload List:

- [operator +\(OracleTimeStampTZ, OracleIntervalDS\)](#)
This static operator adds the supplied OracleIntervalDS to the supplied OracleTimeStampTZ and returns a new OracleTimeStampTZ structure.
- [operator +\(OracleTimeStampTZ, OracleIntervalYM\)](#)
This static operator adds the supplied OracleIntervalYM to the supplied OracleTimeStampTZ and returns a new OracleTimeStampTZ structure.
- [operator +\(OracleTimeStampTZ, TimeSpan\)](#)
This static operator adds the supplied TimeSpan to the supplied OracleTimeStampTZ and returns a new OracleTimeStampTZ structure.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

operator +(OracleTimeStampTZ, OracleIntervalDS)

This static operator adds the supplied OracleIntervalDS to the supplied OracleTimeStampTZ and returns a new OracleTimeStampTZ structure.

Declaration

```
// C#  
public static operator +(OracleTimeStampTZ value1,  
    OracleIntervalDS value2);
```

Parameters

- *value1*
An OracleTimeStampTZ.
- *value2*

An `OracleIntervalDS`.

Return Value

An `OracleTimeStampTZ`.

Remarks

If either parameter has a null value, the returned `OracleTimeStampTZ` has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

operator +(OracleTimeStampTZ, OracleIntervalYM)

This static operator adds the supplied `OracleIntervalYM` to the supplied `OracleTimeStampTZ` and returns a new `OracleTimeStampTZ` structure.

Declaration

```
// C#  
public static operator +(OracleTimeStampTZ value1,  
    OracleIntervalYM value2);
```

Parameters

- *value1*
An `OracleTimeStampTZ`.
- *value2*
An `OracleIntervalYM`.

Return Value

An `OracleTimeStampTZ`.

Remarks

If either parameter has a null value, the returned `OracleTimeStampTZ` has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

operator +(OracleTimeStampTZ, TimeSpan)

This static operator adds the supplied `TimeSpan` to the supplied `OracleTimeStampTZ` and returns a new `OracleTimeStampTZ` structure.

Declaration

```
// C#  
public static operator +(OracleTimeStampTZ value1, TimeSpan value2);
```

Parameters

- *value1*
An `OracleTimeStampTZ`.
- *value2*
A `TimeSpan`.

Return Value

An `OracleTimeStampTZ`.

Remarks

If the `OracleTimeStampTZ` instance has a null value, the returned `OracleTimeStampTZ` has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

operator ==

This static operator determines if two `OracleTimeStampTZ` values are equal.

Declaration

```
// C#  
public static bool operator ==(OracleTimeStampTZ value1,  
    OracleTimeStampTZ value2);
```

Parameters

- *value1*
The first `OracleTimeStampTZ`.
- *value2*
The second `OracleTimeStampTZ`.

Return Value

Returns true if they are equal; otherwise returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleTimeStampTZ that has a value is greater than an OracleTimeStampTZ that has a null value.
- Two OracleTimeStampTZs that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

operator >

This static operator determines if the first of two OracleTimeStampTZ values is greater than the second.

Declaration

```
// C#  
public static bool operator > (OracleTimeStampTZ value1,  
    OracleTimeStampTZ value2);
```

Parameters

- *value1*
The first OracleTimeStampTZ.
- *value2*
The second OracleTimeStampTZ.

Return Value

Returns true if the first OracleTimeStampTZ value is greater than the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleTimeStampTZ that has a value is greater than an OracleTimeStampTZ that has a null value.
- Two OracleTimeStampTZs that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

operator >=

This static operator determines if the first of two `OracleTimeStampTZ` values is greater than or equal to the second.

Declaration

```
// C#  
public static bool operator >= (OracleTimeStampTZ value1,  
    OracleTimeStampTZ value2);
```

Parameters

- *value1*
The first `OracleTimeStampTZ`.
- *value2*
The second `OracleTimeStampTZ`.

Return Value

Returns `true` if the first `OracleTimeStampTZ` is greater than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.
- Two `OracleTimeStampTZ`s that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

operator !=

This static operator determines if two `OracleTimeStampTZ` values are not equal.

Declaration

```
// C#
public static bool operator != (OracleTimeStampTZ value1,
    OracleTimeStampTZ value2);
```

Parameters

- *value1*
The first OracleTimeStampTZ.
- *value2*
The second OracleTimeStampTZ.

Return Value

Returns true if two OracleTimeStampTZ values are not equal; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleTimeStampTZ that has a value is greater than an OracleTimeStampTZ that has a null value.
- Two OracleTimeStampTZs that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

operator <

This static operator determines if the first of two OracleTimeStampTZ values is less than the second.

Declaration

```
// C#
public static bool operator < (OracleTimeStampTZ value1,
    OracleTimeStampTZ value2);
```

Parameters

- *value1*
The first OracleTimeStampTZ.
- *value2*
The second OracleTimeStampTZ.

Return Value

Returns true if the first `OracleTimeStampTZ` is less than the second; otherwise returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.
- Two `OracleTimeStampTZ`s that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

operator <=

This static operator determines if the first of two `OracleTimeStampTZ` values is less than or equal to the second.

Declaration

```
// C#  
public static bool operator <= (OracleTimeStampTZ value1,  
    OracleTimeStampTZ value2);
```

Parameters

- *value1*
The first `OracleTimeStampTZ`.
- *value2*
The second `OracleTimeStampTZ`.

Return Value

Returns true if the first `OracleTimeStampTZ` is less than or equal to the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.
- Two `OracleTimeStampTZ`s that contain a null value are equal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

operator -

operator - subtracts the supplied value, from the supplied OracleTimeStampTZ value, and returns a new OracleTimeStampTZ structure.

Overload List:

- [operator - \(OracleTimeStampTZ, OracleIntervalDS\)](#)
This static operator subtracts the supplied OracleIntervalDS value, from the supplied OracleTimeStampTZ value, and return a new OracleTimeStampTZ structure.
- [operator - \(OracleTimeStampTZ, OracleIntervalYM\)](#)
This static operator subtracts the supplied OracleIntervalYM value, from the supplied OracleTimeStampTZ value, and returns a new OracleTimeStampTZ structure.
- [operator - \(OracleTimeStampTZ value1, TimeSpan value2\)](#)
This static operator subtracts the supplied TimeSpan value, from the supplied OracleTimeStampTZ value, and returns a new OracleTimeStampTZ structure.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

operator - (OracleTimeStampTZ, OracleIntervalDS)

This static operator subtracts the supplied OracleIntervalDS value, from the supplied OracleTimeStampTZ value, and return a new OracleTimeStampTZ structure.

Declaration

```
// C#  
public static operator - (OracleTimeStampTZ value1,  
    OracleIntervalDS value2);
```

Parameters

- *value1*
An OracleTimeStampTZ.
- *value2*

An `OracleIntervalDS`.

Return Value

An `OracleTimeStampTZ` structure.

Remarks

If either parameter has a null value, the returned `OracleTimeStampTZ` has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

operator - (OracleTimeStampTZ, OracleIntervalYM)

This static operator subtracts the supplied `OracleIntervalYM` value, from the supplied `OracleTimeStampTZ` value, and returns a new `OracleTimeStampTZ` structure.

Declaration

```
// C#  
public static operator - (OracleTimeStampTZ value1,  
    OracleIntervalYM value2);
```

Parameters

- *value1*
An `OracleTimeStampTZ`.
- *value2*
An `OracleIntervalYM`.

Return Value

An `OracleTimeStampTZ` structure.

Remarks

If either parameter has a null value, the returned `OracleTimeStampTZ` has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

operator - (OracleTimeStampTZ value1, TimeSpan value2)

This static operator subtracts the supplied TimeSpan value, from the supplied OracleTimeStampTZ value, and returns a new OracleTimeStampTZ structure.

Declaration

```
// C#
public static operator - (OracleTimeStampTZ value1, TimeSpan value2);
```

Parameters

- *value1*
An OracleTimeStampTZ.
- *value2*
A TimeSpan.

Return Value

An OracleTimeStampTZ structure.

Remarks

If the OracleTimeStampTZ instance has a null value, the returned OracleTimeStampTZ structure has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

OracleTimeStampTZ Static Type Conversions

The OracleTimeStampTZ static type conversions are listed in [Table 14-137](#).

Table 14-137 OracleTimeStampTZ Static Type Conversions

Operator	Description
explicit operator OracleTimeStampTZ	Converts an instance value to an OracleTimeStampTZ structure (Overloaded)
implicit operator OracleTimeStampTZ	Converts an instance value to an OracleTimeStampTZ structure (Overloaded)
explicit operator DateTime	Converts an OracleTimeStampTZ value to a DateTime structure in the current time zone

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

explicit operator OracleTimeStampTZ

explicit operator OracleTimeStampTZ converts an instance value to an OracleTimeStampTZ structure.

Overload List:

- [explicit operator OracleTimeStampTZ\(OracleTimeStamp\)](#)
This static type conversion operator converts an OracleTimeStamp value to an OracleTimeStampTZ structure.
- [explicit operator OracleTimeStampTZ\(OracleTimeStampLTZ\)](#)
This static type conversion operator converts an OracleTimeStampLTZ value to an OracleTimeStampTZ structure.
- [explicit operator OracleTimeStampTZ\(string\)](#)
This static type conversion operator converts the supplied string value to an OracleTimeStampTZ structure.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)

explicit operator OracleTimeStampTZ(OracleTimeStamp)

This static type conversion operator converts an OracleTimeStamp value to an OracleTimeStampTZ structure.

Declaration

```
// C#  
public static explicit operator OracleTimeStampTZ(OracleTimeStamp value1);
```

Parameters

- *value1*
An OracleTimeStamp.

Return Value

The returned `OracleTimeStampTZ` contains the date and time from the `OracleTimeStamp` and the time zone from the `OracleGlobalization.TimeZone` of the thread.

Remarks

The `OracleGlobalization.TimeZone` of the thread is used to convert from an `OracleTimeStamp` structure to an `OracleTimeStampTZ` structure.

If the `OracleTimeStamp` structure has a null value, the returned `OracleTimeStampTZ` structure also has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)

explicit operator OracleTimeStampTZ(OracleTimeStampLTZ)

This static type conversion operator converts an `OracleTimeStampLTZ` value to an `OracleTimeStampTZ` structure.

Declaration

```
// C#  
public static explicit operator OracleTimeStampTZ(OracleTimeStampLTZ value1);
```

Parameters

- *value1*
An `OracleTimeStampLTZ`.

Return Value

The returned `OracleTimeStampTZ` contains the date and time from the `OracleTimeStampLTZ` and the time zone from the `OracleGlobalization.TimeZone` of the thread.

Remarks

If the `OracleTimeStampLTZ` structure has a null value, the returned `OracleTimeStampTZ` structure also has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)

explicit operator OracleTimeStampTZ(string)

This static type conversion operator converts the supplied string value to an OracleTimeStampTZ structure.

Declaration

```
// C#  
public static explicit operator OracleTimeStampTZ(string tsStr);
```

Parameters

- *tsStr*

A string representation of an Oracle TIMESTAMP WITH TIME ZONE.

Return Value

An OracleTimeStampTZ value.

Exceptions

ArgumentException - The *tsStr* is an invalid string representation of an Oracle TIMESTAMP WITH TIME ZONE, or the *tsStr* is not in the timestamp format specified by the thread's OracleGlobalization.TimestampTZFormat property, which represents the Oracle NLS_TIMESTAMP_TZ_FORMAT parameter.

Remarks

The names and abbreviations used for months and days are in the language specified by the DateLanguage and Calendar properties of the thread's OracleGlobalization object. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#  
  
using System;  
using Oracle.DataAccess.Client;  
using Oracle.DataAccess.Types;  
  
class OracleTimeStampTZSample  
{  
    static void Main()  
    {  
        // Set the nls_timestamp_tz_format for the explicit operator
```

```
// OracleTimeStampTZ(string)
OracleGlobalization info = OracleGlobalization.GetClientInfo();
info.TimeStampTZFormat = "DD-MON-YYYY HH:MI:SS.FF AM TZR";
OracleGlobalization.SetThreadInfo(info);

// construct OracleTimeStampTZ from a string using the format specified.
OracleTimeStampTZ tstz = new OracleTimeStampTZ("11-NOV-1999" +
    "11:02:33.444 AM US/Pacific");

// Set the nls_timestamp_tz_format for the ToString() method
info.TimeStampTZFormat = "YYYY-MON-DD HH:MI:SS.FF AM TZR";
OracleGlobalization.SetThreadInfo(info);
Console.WriteLine(tstz.ToString());
    }
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)

implicit operator OracleTimeStampTZ

implicit operator OracleTimeStampTZ converts a DateTime structure to an OracleTimeStampTZ structure.

Overload List:

- [implicit operator OracleTimeStampTZ\(OracleDate\)](#)
This static type conversion operator converts an OracleDate value to an OracleTimeStampTZ structure.
- [implicit operator OracleTimeStampTZ\(DateTime\)](#)
This static type conversion operator converts a DateTime structure to an OracleTimeStampTZ structure.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)

implicit operator OracleTimeStampTZ(OracleDate)

This static type conversion operator converts an `OracleDate` value to an `OracleTimeStampTZ` structure.

Declaration

```
// C#  
public static implicit operator OracleTimeStampTZ(OracleDate value1);
```

Parameters

- *value1*
An `OracleDate`.

Return Value

The returned `OracleTimeStampTZ` contains the date and time from the `OracleDate` and the time zone from the `OracleGlobalization.TimeZone` of the thread.

Remarks

The `OracleGlobalization.TimeZone` of the thread is used to convert from an `OracleDate` to an `OracleTimeStampTZ` structure. If the `OracleDate` structure has a null value, the returned `OracleTimeStampTZ` structure also has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)

implicit operator OracleTimeStampTZ(DateTime)

This static type conversion operator converts a `DateTime` structure to an `OracleTimeStampTZ` structure.

Declaration

```
// C#  
public static implicit operator OracleTimeStampTZ (DateTime value1);
```

Parameters

- *value1*
A `DateTime` structure.

Return Value

The returned `OracleTimeStampTZ` contains the date and time from the `DateTime` and the time zone from the `OracleGlobalization.TimeZone` of the thread.

Remarks

The `OracleGlobalization.TimeZone` of the thread is used to convert from a `DateTime` to an `OracleTimeStampTZ` structure.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)

explicit operator DateTime

This static type conversion operator converts an `OracleTimeStampTZ` value to a `DateTime` structure and truncates the time zone information.

Declaration

```
// C#  
public static explicit operator DateTime(OracleTimeStampTZ value1);
```

Parameters

- *value1*
An `OracleTimeStampTZ`.

Return Value

A `DateTime` containing the date and time in the current instance, but with the time zone information in the current instance truncated.

Exceptions

`OracleNullValueException` - The `OracleTimeStampTZ` structure has a null value.

Remarks

The precision of the `OracleTimeStampTZ` value can be lost during the conversion, and the time zone information in the current instance is truncated

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

OracleTimeStampTZ Properties

The OracleTimeStampTZ properties are listed in [Table 14-138](#).

Table 14-138 OracleTimeStampTZ Properties

Properties	Description
BinData	Returns an array of bytes that represents an Oracle TIMESTAMP WITH TIME ZONE in Oracle internal format
Day	Specifies the day component of an OracleTimeStampTZ in the current time zone
IsNull	Indicates whether or not the current instance has a null value
Hour	Specifies the hour component of an OracleTimeStampTZ in the current time zone
Millisecond	Specifies the millisecond component of an OracleTimeStampTZ in the current time zone
Minute	Specifies the minute component of an OracleTimeStampTZ in the current time zone
Month	Specifies the month component of an OracleTimeStampTZ in the current time zone
Nanosecond	Specifies the nanosecond component of an OracleTimeStampTZ in the current time zone
Second	Specifies the second component of an OracleTimeStampTZ in the current time zone
TimeZone	Returns the time zone of the OracleTimeStampTZ instance
Value	Returns the date and time that is stored in the OracleTimeStampTZ structure in the current time zone
Year	Specifies the year component of an OracleTimeStampTZ

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

BinData

This property returns an array of bytes that represents an Oracle TIMESTAMP WITH TIME ZONE in Oracle internal format.

Declaration

```
// C#  
public byte[] BinData {get;}
```

Property Value

The provided byte array that represents an Oracle TIMESTAMP WITH TIME ZONE in Oracle internal format.

Exceptions

OracleNullValueException - The current instance has a null value.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

Day

This property specifies the day component of an OracleTimeStampTZ in the current time zone.

Declaration

```
// C#  
public int Day{get;}
```

Property Value

A number that represents the day. Range of Day is (1 to 31).

Exceptions

OracleNullValueException - The current instance has a null value.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

IsNull

This property indicates whether or not the current instance has a null value.

Declaration

```
// C#  
public bool IsNull{get;}
```

Property Value

Returns true if the current instance has a null value. Otherwise, returns false.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

Hour

This property specifies the hour component of an OracleTimeStampTZ in the current time zone.

Declaration

```
// C#  
public int Hour{get;}
```

Property Value

A number that represents the hour. Range of Hour is (0 to 23).

Exceptions

OracleNullValueException - The current instance has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

Millisecond

This property gets the millisecond component of an OracleTimeStampTZ in the current time zone.

Declaration

```
// C#  
public double Millisecond{get;}
```

Property Value

A number that represents a millisecond. Range of Millisecond is (0 to 999.999999)

Exceptions

OracleNullValueException - The current instance has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

Minute

This property gets the minute component of an OracleTimeStampTZ in the current time zone.

Declaration

```
// C#  
public int Minute{get;}
```

Property Value

A number that represent a minute. Range of Minute is (0 to 59).

Exceptions

OracleNullValueException - The current instance has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

Month

This property gets the month component of an OracleTimeStampTZ in the current time zone

Declaration

```
// C#  
public int Month{get;}
```

Property Value

A number that represents a month. Range of Month is (1 to 12).

Exceptions

OracleNullValueException - The current instance has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

Nanosecond

This property gets the nanosecond component of an OracleTimeStampTZ in the current time zone.

Declaration

```
// C#  
public int Nanosecond{get;}
```

Property Value

A number that represents a nanosecond. Range of Nanosecond is (0 to 999999999).

Exceptions

OracleNullValueException - The current instance has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

Second

This property gets the second component of an OracleTimeStampTZ in the current time zone.

Declaration

```
// C#  
public int Second{get;}
```

Property Value

A number that represents a second. Range of Second is (0 to 59).

Exceptions

OracleNullValueException - The current instance has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

TimeZone

This property returns the time zone of the OracleTimeStampTZ instance.

Declaration

```
// C#  
public string TimeZone{get;}
```

Property Value

A string that represents the time zone.

Remarks

If no time zone is specified in the constructor, this property is set to the thread's OracleGlobalization.TimeZone by default

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)

Value

This property returns the date and time that is stored in the OracleTimeStampTZ structure in the current time zone.

Declaration

```
// C#  
public DateTime Value{get;}
```

Property Value

A DateTime in the current time zone.

Exceptions

OracleNullValueException - The current instance has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

Year

This property sets the year component of an OracleTimeStampTZ in the current time zone.

Declaration

```
// C#
public int Year{get;}
```

Property Value

A number that represents a year. The range of Year is (-4712 to 9999).

Exceptions

OracleNullValueException - The current instance has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

OracleTimeStampTZ Methods

The OracleTimeStampTZ methods are listed in [Table 14-139](#).

Table 14-139 OracleTimeStampTZ Methods

Methods	Description
AddDays	Adds the supplied number of days to the current instance
AddHours	Adds the supplied number of hours to the current instance

Table 14-139 (Cont.) OracleTimeStampTZ Methods

Methods	Description
AddMilliseconds	Adds the supplied number of milliseconds to the current instance
AddMinutes	Adds the supplied number of minutes to the current instance
AddMonths	Adds the supplied number of months to the current instance
AddNanoseconds	Adds the supplied number of nanoseconds to the current instance
AddSeconds	Adds the supplied number of seconds to the current instance
AddYears	Adds the supplied number of years to the current instance
CompareTo	Compares the current OracleTimeStampTZ instance to an object, and returns an integer that represents their relative values
Equals	Determines whether or not an object has the same date and time as the current OracleTimeStampTZ instance (Overloaded)
GetDaysBetween	Subtracts an OracleTimeStampTZ from the current instance and returns an OracleIntervalDS that represents the time interval
GetHashCode	Returns a hash code for the OracleTimeStampTZ instance
GetTimeZoneOffset	Gets the time zone information in hours and minutes of the current OracleTimeStampTZ
GetYearsBetween	Subtracts an OracleTimeStampTZ from the current instance and returns an OracleIntervalYM that represents the time interval
GetType	Inherited from System.Object
ToLocalTime	Converts the current OracleTimeStampTZ instance to local time
ToOracleDate	Converts the current OracleTimeStampTZ structure to an OracleDate structure
ToOracleTimeStampLTZ	Converts the current OracleTimeStampTZ structure to an OracleTimeStampLTZ structure
ToOracleTimeStamp	Converts the current OracleTimeStampTZ structure to an OracleTimeStamp structure
ToString	Converts the current OracleTimeStampTZ structure to a string
ToUniversalTime	Converts the current datetime to Coordinated Universal Time (UTC)

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

AddDays

This method adds the supplied number of days to the current instance.

Declaration

```
// C#  
public OracleTimeStampTZ AddDays(double days);
```

Parameters

- *days*

The supplied number of days. Range is $(-1,000,000,000 < days < 1,000,000,000)$

Return Value

An `OracleTimeStampTZ`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

`ArgumentOutOfRangeException` - The argument value is out of the specified range.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

AddHours

This method adds the supplied number of hours to the current instance.

Declaration

```
// C#  
public OracleTimeStampTZ AddHours(double hours);
```

Parameters

- *hours*

The supplied number of hours. Range is $(-24,000,000,000 < hours < 24,000,000,000)$.

Return Value

An `OracleTimeStampTZ`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

`ArgumentOutOfRangeException` - The argument value is out of the specified range.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

AddMilliseconds

This method adds the supplied number of milliseconds to the current instance.

Declaration

```
// C#  
public OracleTimeStampTZ AddMilliseconds(double milliseconds);
```

Parameters

- *milliseconds*

The supplied number of milliseconds. Range is $(-8.64 * 10^{16} < \text{milliseconds} < 8.64 * 10^{16})$.

Return Value

An OracleTimeStampTZ.

Exceptions

OracleNullValueException - The current instance has a null value.

ArgumentOutOfRangeException - The argument value is out of the specified range.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

AddMinutes

This method adds the supplied number of minutes to the current instance.

Declaration

```
// C#  
public OracleTimeStampTZ AddMinutes(double minutes);
```

Parameters

- *minutes*

The supplied number of minutes. Range is $(-1,440,000,000,000 < minutes < 1,440,000,000,000)$.

Return Value

An OracleTimeStampTZ.

Exceptions

OracleNullValueException - The current instance has a null value.

ArgumentOutOfRangeException - The argument value is out of the specified range.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

AddMonths

This method adds the supplied number of months to the current instance.

Declaration

```
// C#  
public OracleTimeStampTZ AddMonths(long months);
```

Parameters

- *months*

The supplied number of months. Range is $(-12,000,000,000 < months < 12,000,000,000)$.

Return Value

An OracleTimeStampTZ.

Exceptions

OracleNullValueException - The current instance has a null value.

ArgumentOutOfRangeException - The argument value is out of the specified range.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

AddNanoseconds

This method adds the supplied number of nanoseconds to the current instance.

Declaration

```
// C#  
public OracleTimeStampTZ AddNanoseconds(long nanoseconds);
```

Parameters

- *nanoseconds*
The supplied number of nanoseconds.

Return Value

An `OracleTimeStampTZ`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

AddSeconds

This method adds the supplied number of seconds to the current instance.

Declaration

```
// C#  
public OracleTimeStampTZ AddSeconds(double seconds);
```

Parameters

- *seconds*
The supplied number of seconds. Range is $(-8.64 * 10^{13} < \text{seconds} < 8.64 * 10^{13})$.

Return Value

An `OracleTimeStampTZ`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

`ArgumentOutOfRangeException` - The argument value is out of the specified range.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

AddYears

This method adds the supplied number of years to the current instance

Declaration

```
// C#  
public OracleTimeStampTZ AddYears(int years);
```

Parameters

- *years*
The supplied number of years. Range is $(-999,999,999 \leq \text{years} \leq 999,999,999)$.

Return Value

An OracleTimeStampTZ.

Exceptions

OracleNullValueException - The current instance has a null value.

ArgumentOutOfRangeException - The argument value is out of the specified range.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

CompareTo

This method compares the current OracleTimeStampTZ instance to an object, and returns an integer that represents their relative values.

Declaration

```
// C#  
public int CompareTo(object obj);
```

Parameters

- *obj*
The object being compared to the current OracleTimeStampTZ instance.

Return Value

The method returns a number that is:

Less than zero: if the current `OracleTimeStampTZ` instance value is less than that of *obj*.

Zero: if the current `OracleTimeStampTZ` instance and *obj* values are equal.

Greater than zero: if the current `OracleTimeStampTZ` instance value is greater than that of *obj*.

Implements

`Comparable`

Exceptions

`ArgumentException` - The *obj* is not of type `OracleTimeStampTZ`.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between `OracleTimeStampTZ`s. For example, comparing an `OracleTimeStampTZ` instance with an `OracleBinary` instance is not allowed. When an `OracleTimeStampTZ` is compared with a different type, an `ArgumentException` is thrown.
- Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.
- Two `OracleTimeStampTZ`s that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

Equals

Overrides `Object`

This method determines whether or not an object has the same date and time as the current `OracleTimeStampTZ` instance.

Declaration

```
// C#  
public override bool Equals(object obj);
```

Parameters

- *obj*

The object being compared to the current `OracleTimeStampTZ` instance.

Return Value

Returns true if the *obj* is of type `OracleTimeStampTZ` and represents the same date and time; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.
- Two `OracleTimeStampTZ`s that contain a null value are equal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

GetDaysBetween

This method subtracts an `OracleTimeStampTZ` value from the current instance and returns an `OracleIntervalDS` that represents the time interval.

Declaration

```
// C#  
public OracleIntervalDS GetDaysBetween(OracleTimeStampTZ value1);
```

Parameters

- *value1*
The `OracleTimeStampTZ` value being subtracted.

Return Value

An `OracleIntervalDS` that represents the interval between two `OracleTimeStampTZ` values.

Remarks

If either the current instance or the parameter has a null value, the returned `OracleIntervalDS` has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

GetHashCode

Overrides Object

This method returns a hash code for the OracleTimeStampTZ instance.

Declaration

```
// C#  
public override int GetHashCode();
```

Return Value

A number that represents the hash code.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

GetTimeZoneOffset

This method gets the time zone portion in hours and minutes of the current OracleTimeStampTZ.

Declaration

```
// C#  
public TimeSpan GetTimeZoneOffset();
```

Return Value

A TimeSpan.

Exceptions

OracleNullValueException - The current instance has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

GetYearsBetween

This method subtracts an OracleTimeStampTZ value from the current instance and returns an OracleIntervalYM that represents the time interval.

Declaration

```
// C#  
public OracleIntervalYM GetYearsBetween(OracleTimeStampTZ val);
```

Parameters

- `val`

The `OracleTimeStampTZ` value being subtracted.

Return Value

An `OracleIntervalYM` that represents the interval between two `OracleTimeStampTZ` values.

Remarks

If either the current instance or the parameter has a null value, the returned `OracleIntervalYM` has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

ToLocalTime

This method converts the current `OracleTimeStampTZ` instance to local time.

Declaration

```
// C#  
public OracleTimeStampLTZ ToLocalTime();
```

Return Value

An `OracleTimeStampLTZ` that contains the date and time, which is normalized to the client local time zone, in the current instance.

Remarks

If the current instance has a null value, the returned `OracleTimeStampLTZ` has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

ToOracleDate

This method converts the current `OracleTimeStampTZ` structure to an `OracleDate` structure.

Declaration

```
// C#  
public OracleDate ToOracleDate();
```

Return Value

The returned `OracleDate` contains the date and time in the current instance, but the time zone information in the current instance is truncated.

Remarks

The precision of the `OracleTimeStampTZ` value can be lost during the conversion, and the time zone information in the current instance is truncated.

If the current instance has a null value, the value of the returned `OracleDate` structure has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

ToOracleTimeStampLTZ

This method converts the current `OracleTimeStampTZ` structure to an `OracleTimeStampLTZ` structure.

Declaration

```
// C#  
public OracleTimeStampLTZ ToOracleTimeStampLTZ();
```

Return Value

The returned `OracleTimeStampLTZ` structure contains the date and time, which is normalized to the client local time zone, in the current instance.

Remarks

If the value of the current instance has a null value, the value of the returned `OracleTimeStampLTZ` structure has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

ToOracleTimeStamp

This method converts the current `OracleTimeStampTZ` structure to an `OracleTimeStamp` structure.

Declaration

```
// C#  
public OracleTimeStamp ToOracleTimeStamp();
```

Return Value

The returned `OracleTimeStamp` contains the date and time in the current instance, but the time zone information is truncated.

Remarks

If the value of the current instance has a null value, the value of the returned `OracleTimeStamp` structure has a null value.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

ToString

Overrides `Object`

This method converts the current `OracleTimeStampTZ` structure to a string.

Declaration

```
// C#  
public override string ToString();
```

Return Value

A string that represents the same date and time as the current `OracleTimeStampTZ` structure.

Remarks

The returned value is a string representation of an `OracleTimeStampTZ` in the format specified by the `OracleGlobalization.TimeStampTZFormat` property of the thread. The names and

abbreviations used for months and days are in the language specified by the `OracleGlobalization.DateLanguage` and the `OracleGlobalization.Calendar` properties of the thread. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class ToStringSample
{
    static void Main()
    {
        // Set the nls parameters for the current thread
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.TimeZone = "US/Eastern";
        info.TimestampFormat = "DD-MON-YYYY HH:MI:SS.FF AM";
        info.TimestampTZFormat = "DD-MON-YYYY HH:MI:SS.FF AM TZR";
        OracleGlobalization.SetThreadInfo(info);

        // Create an OracleTimeStampTZ in US/Pacific time zone
        OracleTimeStampTZ tstz1=new OracleTimeStampTZ("11-NOV-1999 "+"
            "11:02:33.444 AM US/Pacific");

        // Note that ToOracleTimeStampTZ uses the thread's time zone region,
        // "US/Eastern"
        OracleTimeStamp ts = new OracleTimeStamp("11-NOV-1999 11:02:33.444 AM");
        OracleTimeStampTZ tstz2 = ts.ToOracleTimeStampTZ();

        // Calculate the difference between tstz1 and tstz2
        OracleIntervalDS idsDiff = tstz1.GetDaysBetween(tstz2);

        // Prints "US/Pacific"
        Console.WriteLine("tstz1.TimeZone = " + tstz1.TimeZone);

        // Prints "US/Eastern"
        Console.WriteLine("tstz2.TimeZone = " + tstz2.TimeZone);

        // Prints 3
        Console.WriteLine("idsDiff.Hours = " + idsDiff.Hours);

        // Prints 0
        Console.WriteLine("idsDiff.Minutes = " + idsDiff.Minutes);
    }
}
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)
- ["OracleGlobalization Class"](#)
- ["Globalization Support "](#)

ToUniversalTime

This method converts the current datetime to Coordinated Universal Time (UTC).

Declaration

```
// C#  
public OracleTimeStampTZ ToUniversalTime();
```

Return Value

An `OracleTimeStampTZ` structure.

Remarks

If the current instance has a null value, the value of the returned `OracleTimeStampTZ` structure has a null value.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTimeStampTZ Structure](#)
- [OracleTimeStampTZ Members](#)

INullable Interface

The `INullable` interface is used to determine whether or not an ODP.NET type has a `NULL` value.

Declaration

```
// C#  
public interface INullable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [INullable Interface Members](#)
- [INullable Interface Properties](#)

INullable Interface Members

INullable members are listed in the following tables.

INullable Interface Properties

INullable interface properties are listed in [Table 14-140](#).

Table 14-140 INullable Interface Properties

Public Property	Description
IsNull	Indicates whether or not the ODP.NET type has a NULL value

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [INullable Interface](#)

INullable Interface Properties

INullable interface properties are listed in [Table 14-140](#).

Table 14-141 INullable Interface Properties

Public Property	Description
IsNull	Indicates whether or not the ODP.NET type has a NULL value

IsNull

This property indicates whether or not the ODP.NET type has a NULL value.

Declaration

```
// C#  
bool IsNull {get;}
```

Property Value

Returns true if the ODP.NET type has a NULL value; otherwise, returns false.

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [INullable Interface](#)
- [INullable Interface Members](#)

Oracle Data Provider for .NET Types Exceptions

This section covers the ODP.NET Types exceptions.

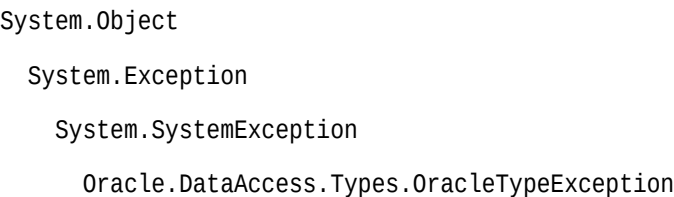
This chapter contains these topics:

- [OracleTypeException Class](#)
- [OracleNullValueException Class](#)
- [OracleTruncateException Class](#)

OracleTypeException Class

The `OracleTypeException` is the base exception class for handling exceptions that occur in the ODP.NET Types classes.

Class Inheritance



Declaration

```
// C#
public class OracleTypeException : SystemException
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTypeException Members](#)
- [OracleTypeException Constructors](#)
- [OracleTypeException Static Methods](#)
- [OracleTypeException Properties](#)
- [OracleTypeException Methods](#)

OracleTypeException Members

OracleTypeException members are listed in the following tables.

OracleTypeException Constructors

The OracleTypeException constructors are listed in [Table 15-1](#).

Table 15-1 OracleTypeException Constructor

Constructor	Description
OracleTypeException Constructors	Creates a new instance of the OracleTypeException class (Overloaded)

OracleTypeException Static Methods

The OracleTypeException static methods are listed in [Table 15-2](#).

Table 15-2 OracleTypeException Static Methods

Methods	Description
Equals	Inherited from System.Object (Overloaded)

OracleTypeException Properties

The OracleTypeException properties are listed in [Table 15-3](#).

Table 15-3 OracleTypeException Properties

Properties	Description
HelpLink	Inherited from System.SystemException.Exception
InnerException	Inherited from System.SystemException.Exception
Message	Specifies the error messages that occur in the exception
Number	Specifies the error number that occurs in the exception
Source	Specifies the name of the data provider that generates the error
StackTrace	Inherited from System.SystemException.Exception

Table 15-3 (Cont.) OracleTypeException Properties

Properties	Description
TargetSite	Inherited from System.SystemException.Exception

OracleTypeException Methods

The OracleTypeException methods are listed in [Table 15-4](#).

Table 15-4 OracleTypeException Methods

Methods	Description
Equals	Inherited from System.Object (Overloaded)
GetBaseException	Inherited from System.SystemException.Exception
GetHashCode	Inherited from System.Object
GetObjectData	Inherited from System.SystemException.Exception
GetType	Inherited from System.Object
ToString	Returns the fully qualified name of this exception

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTypeException Class](#)

OracleTypeException Constructors

The OracleTypeException constructors create new instances of the OracleTypeException class.

Overload List:

- [OracleTypeException\(string\)](#)
This constructor creates a new instance of the OracleTypeException class with the specified error message, errMessage.
- [OracleTypeException\(SerializationInfo, StreamingContext\)](#)
This constructor creates a new instance of the OracleTypeException class with the specified serialization information, si, and the specified streaming context, sc.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTypeException Class](#)
- [OracleTypeException Members](#)

OracleTypeException(string)

This constructor creates a new instance of the `OracleTypeException` class with the specified error message, `errorMessage`.

Declaration

```
// C#  
public OracleTypeException (string errorMessage);
```

Parameters

- *errorMessage*
The specified error message.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTypeException Class](#)
- [OracleTypeException Members](#)

OracleTypeException(SerializationInfo, StreamingContext)

This constructor creates a new instance of the `OracleTypeException` class with the specified serialization information, `si`, and the specified streaming context, `sc`.

Declaration

```
// C#  
protected OracleTypeException (SerializationInfo si, StreamingContext sc);
```

Parameters

- *si*
The specified serialization information.
- *sc*
The specified streaming context.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTypeException Class](#)
- [OracleTypeException Members](#)

OracleTypeException Static Methods

The OracleTypeException static methods are listed in [Table 15-5](#).

Table 15-5 OracleTypeException Static Methods

Methods	Description
Equals	Inherited from System.Object (Overloaded)

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTypeException Class](#)
- [OracleTypeException Members](#)

OracleTypeException Properties

The OracleTypeException properties are listed in [Table 15-6](#).

Table 15-6 OracleTypeException Properties

Properties	Description
HelpLink	Inherited from System.SystemException.Exception
InnerException	Inherited from System.SystemException.Exception
Message	Specifies the error messages that occur in the exception
Number	Specifies the error number that occurs in the exception
Source	Specifies the name of the data provider that generates the error
StackTrace	Inherited from System.SystemException.Exception
TargetSite	Inherited from System.SystemException.Exception

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTypeException Class](#)
- [OracleTypeException Members](#)

Message

Overrides Exception

This property specifies the error messages that occur in the exception.

Declaration

```
// C#  
public override string Message {get;}
```

Property Value

An error message.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTypeException Class](#)
- [OracleTypeException Members](#)

Number

Overrides Exception

This property specifies the error number that occurs in the exception

Declaration

```
// C#  
public override int Number {get;}
```

Property Value

An error number

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTypeException Class](#)
- [OracleTypeException Members](#)

Source

Overrides Exception

This property specifies the name of the data provider that generates the error.

Declaration

```
// C#
public override string Source {get;}
```

Property Value

Oracle Data Provider for .NET.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTypeException Class](#)
- [OracleTypeException Members](#)

OracleTypeException Methods

The OracleTypeException methods are listed in [Table 15-7](#).

Table 15-7 OracleTypeException Methods

Methods	Description
Equals	Inherited from System.Object (Overloaded)
GetBaseException	Inherited from System.SystemException.Exception
GetHashCode	Inherited from System.Object
GetObjectData	Inherited from System.SystemException.Exception
GetType	Inherited from System.Object
ToString	Returns the fully qualified name of this exception

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTypeException Class](#)
- [OracleTypeException Members](#)

ToString

Overrides Exception

This method returns the fully qualified name of this exception, the error message in the Message property, the InnerException.ToString() message, and the stack trace.

Declaration

```
// C#
public override string ToString();
```

Return Value

The fully qualified name of this exception.

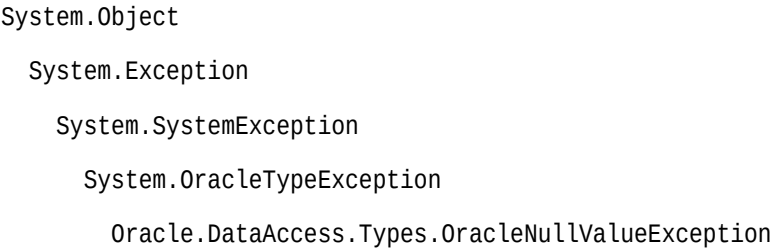
See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTypeException Class](#)
- [OracleTypeException Members](#)

OracleNullValueException Class

The OracleNullValueException represents an exception that is thrown when trying to access an ODP.NET Types structure that has a null value.

Class Inheritance



Declaration

```
// C#
public sealed class OracleNullValueException : OracleTypeException
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleNullValueException Members](#)
- [OracleNullValueException Constructors](#)
- [OracleNullValueException Static Methods](#)
- [OracleNullValueException Properties](#)
- [OracleNullValueException Methods](#)

OracleNullValueException Members

OracleNullValueException members are listed in the following tables.

OracleNullValueException Constructors

The OracleNullValueException constructors are listed in [Table 15-8](#).

Table 15-8 OracleNullValueException Constructors

Constructor	Description
OracleNullValueException Constructors	Creates a new instance of the OracleNullValueException class (Overloaded)

OracleNullValueException Static Methods

The OracleNullValueException static methods are listed in [Table 15-9](#).

Table 15-9 OracleNullValueException Static Methods

Methods	Description
Equals	Inherited from System.Object (Overloaded)

OracleNullValueException Properties

The OracleNullValueException properties are listed in [Table 15-10](#).

Table 15-10 OracleNullValueException Properties

Properties	Description
HelpLink	Inherited from System.SystemException.Exception
InnerException	Inherited from System.SystemException.Exception

Table 15-10 (Cont.) OracleNullValueException Properties

Properties	Description
Message	Inherited from OracleTypeException
Source	Inherited from OracleTypeException
StackTrace	Inherited from System.SystemException.Exception
TargetSite	Inherited from System.SystemException.Exception

OracleNullValueException Methods

The OracleNullValueException methods are listed in [Table 15-11](#).

Table 15-11 OracleNullValueException Methods

Methods	Description
Equals	Inherited from System.Object (Overloaded)
GetBaseException	Inherited from System.SystemException.Exception
GetHashCode	Inherited from System.Object
GetObjectData	Inherited from System.SystemException.Exception
GetType	Inherited from System.Object
ToString	Inherited from OracleTypeException

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleNullValueException Class](#)

OracleNullValueException Constructors

The OracleNullValueException constructors create new instances of the OracleNullValueException class.

Overload List:

- [OracleNullValueException\(\)](#)

This constructor creates a new instance of the OracleNullValueException class with its default properties.

- [OracleNullValueException\(string\)](#)

This constructor creates a new instance of the OracleNullValueException class with the specified error message, `errorMessage`.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleNullValueException Class](#)
- [OracleNullValueException Members](#)

OracleNullValueException()

This constructor creates a new instance of the `OracleNullValueException` class with its default properties.

Declaration

```
// C#  
public OracleNullValueException();
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleNullValueException Class](#)
- [OracleNullValueException Members](#)

OracleNullValueException(string)

This constructor creates a new instance of the `OracleNullValueException` class with the specified error message, `errorMessage`.

Declaration

```
// C#  
public OracleNullValueException (string errorMessage);
```

Parameters

- *errorMessage*
The specified error message.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleNullValueException Class](#)
- [OracleNullValueException Members](#)

OracleNullValueException Static Methods

The OracleNullValueException static methods are listed in [Table 15-12](#).

Table 15-12 OracleNullValueException Static Methods

Methods	Description
Equals	Inherited from System.Object (Overloaded)

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleNullValueException Class](#)
- [OracleNullValueException Members](#)

OracleNullValueException Properties

The OracleNullValueException properties are listed in [Table 15-13](#).

Table 15-13 OracleNullValueException Properties

Properties	Description
HelpLink	Inherited from System.SystemException.Exception
InnerException	Inherited from System.SystemException.Exception
Message	Inherited from OracleTypeException
Source	Inherited from OracleTypeException
StackTrace	Inherited from System.SystemException.Exception
TargetSite	Inherited from System.SystemException.Exception

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleNullValueException Class](#)
- [OracleNullValueException Members](#)

OracleNullValueException Methods

The OracleNullValueException methods are listed in [Table 15-14](#).

Table 15-14 OracleNullValueException Methods

Methods	Description
Equals	Inherited from System.Object (Overloaded)
GetBaseException	Inherited from System.SystemException.Exception
GetHashCode	Inherited from System.Object
GetObjectData	Inherited from System.SystemException.Exception
GetType	Inherited from System.Object
ToString	Inherited from OracleTypeException

OracleTruncateException Class

The OracleTruncateException class represents an exception that is thrown when truncation in a ODP.NET Types class occurs.

Class Inheritance

System.Object

System.Exception

System.SystemException

System.OracleTypeException

Oracle.DataAccess.Types.OracleTruncateException

Declaration

```
// C#
public sealed class OracleTruncateException : OracleTypeException
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTruncateException Members](#)
- [OracleTruncateException Constructors](#)
- [OracleTruncateException Static Methods](#)
- [OracleTruncateException Properties](#)
- [OracleTruncateException Methods](#)

OracleTruncateException Members

OracleTruncateException members are listed in the following tables.

OracleTruncateException Constructors

The OracleTruncateException constructors are listed in [Table 15-15](#).

Table 15-15 OracleTruncateException Constructors

Constructor	Description
OracleTruncateException Constructors	Creates a new instance of the OracleTruncateException class (Overloaded)

OracleTruncateException Static Methods

The OracleTruncateException static methods are listed in [Table 15-16](#).

Table 15-16 OracleTruncateException Static Methods

Methods	Description
Equals	Inherited from System.Object (Overloaded)

OracleTruncateException Properties

The OracleTruncateException properties are listed in [Table 15-17](#).

Table 15-17 OracleTruncateException Properties

Properties	Description
HelpLink	Inherited from System.SystemException.Exception
InnerException	Inherited from System.SystemException.Exception
Message	Inherited from OracleTypeException
Source	Inherited from OracleTypeException
StackTrace	Inherited from System.SystemException.Exception
TargetSite	Inherited from System.SystemException.Exception

OracleTruncateException Methods

The OracleTruncateException methods are listed in [Table 15-18](#).

Table 15-18 OracleTruncateException Methods

Methods	Description
Equals	Inherited from System.Object (Overloaded)
GetBaseException	Inherited from System.SystemException.Exception
GetHashCode	Inherited from System.Object
GetObjectData	Inherited from System.SystemException.Exception
GetType	Inherited from System.Object
ToString	Inherited from OracleTypeException

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleTruncateException Class](#)

OracleTruncateException Constructors

The OracleTruncateException constructors create new instances of the OracleTruncateException class

Overload List:

- [OracleTruncateException\(\)](#)
This constructor creates a new instance of the OracleTruncateException class with its default properties.
- [OracleTruncateException\(string\)](#)
This constructor creates a new instance of the OracleTruncateException class with the specified error message, errMessage.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleTruncateException Class](#)
- [OracleTruncateException Members](#)

OracleTruncateException()

This constructor creates a new instance of the OracleTruncateException class with its default properties.

Declaration

```
// C#  
public OracleTruncateException();
```

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTruncateException Class](#)
- [OracleTruncateException Members](#)

OracleTruncateException(string)

This constructor creates a new instance of the OracleTruncateException class with the specified error message, `errMessage`.

Declaration

```
// C#  
public OracleTruncateException (string errMessage);
```

Parameters

- *errMessage*
The specified error message.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTruncateException Class](#)
- [OracleTruncateException Members](#)

OracleTruncateException Static Methods

The OracleTruncateException static methods are listed in [Table 15-19](#).

Table 15-19 OracleTruncateException Static Methods

Methods	Description
Equals	Inherited from System.Object (Overloaded)

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTruncateException Class](#)
- [OracleTruncateException Members](#)

OracleTruncateException Properties

The OracleTruncateException properties are listed in [Table 15-20](#).

Table 15-20 OracleTruncateException Properties

Properties	Description
HelpLink	Inherited from System.SystemException.Exception
InnerException	Inherited from System.SystemException.Exception
Message	Inherited from OracleTypeException
Source	Inherited from OracleTypeException
StackTrace	Inherited from System.SystemException.Exception
TargetSite	Inherited from System.SystemException.Exception

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTruncateException Class](#)
- [OracleTruncateException Members](#)

OracleTruncateException Methods

The OracleTruncateException methods are listed in [Table 15-21](#).

Table 15-21 OracleTruncateException Methods

Methods	Description
Equals	Inherited from System.Object (Overloaded)
GetBaseException	Inherited from System.SystemException.Exception
GetHashCode	Inherited from System.Object
GetObjectData	Inherited from System.SystemException.Exception
GetType	Inherited from System.Object
ToString	Inherited from OracleTypeException

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleTruncateException Class](#)
- [OracleTruncateException Members](#)

16

Oracle Data Provider for .NET UDT-Related Classes

This chapter describes the object-related classes and interfaces in the Oracle Data Provider for .NET that provide support for Oracle user-defined data types (UDT).

In ODAC Oracle Universal Installer, samples are provided in the `ORACLE_BASE\ORACLE_HOME\ODP.NET\Samples\UDT` directory.

See Also

["Oracle User-Defined Types \(UDTs\) and .NET Custom Types"](#)

- [OracleCustomTypeMappingAttribute Class](#)
- [OracleObjectMappingAttribute Class](#)
- [OracleArrayMappingAttribute Class](#)
- [IOracleCustomType Interface](#)
- [IOracleCustomTypeFactory Interface](#)
- [IOracleArrayTypeFactory Interface](#)
- [OracleUdt Class](#)
- [OracleRef Class](#)
- [OracleUdtFetchOption Enumeration](#)
- [OracleUdtStatus Enumeration](#)

OracleCustomTypeMappingAttribute Class

The `OracleCustomTypeMappingAttribute` class is used to mark a custom type factory class or struct with information that is used by ODP.NET when a custom type is used to represent an Oracle UDT.

Class Inheritance

`System.Object`

`System.Attribute`

`System.OracleCustomTypeMappingAttribute`

Declaration

```
// C#
[AttributeUsageAttribute(AttributeTargets.Class|AttributeTargets.Struct,
    AllowMultiple=false, Inherited=true)]
public sealed class OracleCustomTypeMappingAttribute : Attribute
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Remarks

The `OracleCustomTypeMapping` attribute must be specified on the custom type factory class to indicate the Oracle UDT that the corresponding custom type represents. The Oracle UDT may be specified in the form `schema_name.type_name`.

For each Oracle UDT that the application uses, there must be a unique custom type factory, as follows:

- Oracle Object Types
The custom type factory must return a custom type that cannot be used to represent any other Oracle Object Type.
- Oracle Collection Types
The custom type factory may return a custom type that can be used by other Oracle Collection Types. This is common when an array type is used to represent an Oracle Collection, that is, when an `int[]` is used to represent a collection of NUMBERS.

If the `OracleCustomTypeMappingAttribute` is not specified, then custom type mappings must be specified through an XML configuration file, for example, `app.config` for Windows applications or the `web.config` for web applications, and the `machine.config`

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleCustomTypeMappingAttribute Members](#)
- [OracleCustomTypeMappingAttribute Constructors](#)
- [OracleCustomTypeMappingAttribute Static Methods](#)
- [OracleCustomTypeMappingAttribute Methods](#)

OracleCustomTypeMappingAttribute Members

OracleCustomTypeMappingAttribute members are listed in the following tables.

OracleCustomTypeMappingAttribute Constructors

OracleCustomTypeMappingAttribute constructors are listed in [Table 16-1](#).

Table 16-1 OracleCustomTypeMappingAttribute Constructors

Constructor	Description
OracleCustomTypeMappingAttribute Constructors	Instantiates a new instance of OracleCustomTypeMappingAttribute class

OracleCustomTypeMappingAttribute Static Methods

OracleCustomTypeMappingAttribute static methods are listed in [Table 16-2](#).

Table 16-2 OracleCustomTypeMappingAttribute Static Methods

Method	Description
Equals	Inherited from System.Attribute
GetCustomAttribute	Inherited from System.Attribute
GetCustomAttributes	Inherited from System.Attribute
IsDefined	Inherited from System.Attribute
ReferenceEquals	Inherited from System.Attribute

OracleCustomTypeMappingAttribute Properties

OracleCustomTypeMappingAttribute properties are listed in [Table 16-3](#).

Table 16-3 OracleCustomTypeMappingAttribute Properties

Property	Description
UdtTypeName	Specifies the Oracle user-defined type name that the custom class maps to
TypeId	Inherited from System.Attribute

OracleCustomTypeMappingAttribute Methods

OracleCustomTypeMappingAttribute methods are listed in [Table 16-4](#).

Table 16-4 OracleCustomTypeMappingAttribute Methods

Method	Description
Equals	Inherited from System.Attribute
GetHashCode	Inherited from System.Attribute
GetType	Inherited from System.Attribute

Table 16-4 (Cont.) OracleCustomTypeMappingAttribute Methods

Method	Description
IsDefaultAttribute	Inherited from System.Attribute
Match	Inherited from System.Attribute
ToString	Inherited from System.Attribute

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleCustomTypeMappingAttribute Class](#)

OracleCustomTypeMappingAttribute Constructors

OracleCustomTypeMappingAttribute constructors create new instances of the OracleCustomTypeMappingAttribute class.

Overload List:

- [OracleCustomTypeMappingAttribute\(string\)](#)

This constructor creates and initializes an OracleCustomTypeMappingAttribute using the specified Oracle user-defined type name.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleCustomTypeMappingAttribute Class](#)
- [OracleCustomTypeMappingAttribute Methods](#)

OracleCustomTypeMappingAttribute(string)

This constructor creates and initializes an OracleCustomTypeMappingAttribute using the specified Oracle user-defined type name.

Declaration

```
// C#  
public OracleCustomTypeMappingAttribute(string udtTypeName)
```

Parameters

- *udtTypeName*

The Oracle user-defined type name that the custom class maps to.

Remarks

The *udtTypeName* parameter is case-sensitive. The *udtTypeName* is specified in the form of *schema_name.type_name*.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleCustomTypeMappingAttribute Class](#)
- [OracleCustomTypeMappingAttribute Members](#)

OracleCustomTypeMappingAttribute Static Methods

OracleCustomTypeMappingAttribute static methods are listed in [Table 16-5](#).

Table 16-5 OracleCustomTypeMappingAttribute Static Methods

Method	Description
Equals	Inherited from System.Attribute
GetCustomAttribute	Inherited from System.Attribute
GetCustomAttributes	Inherited from System.Attribute
IsDefined	Inherited from System.Attribute
ReferenceEquals	Inherited from System.Attribute

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleCustomTypeMappingAttribute Class](#)
- [OracleCustomTypeMappingAttribute Members](#)

OracleCustomTypeMappingAttribute Properties

OracleCustomTypeMappingAttribute properties are listed in [Table 16-6](#).

Table 16-6 OracleCustomTypeMappingAttribute Properties

Property	Description
UdtTypeName	Specifies the Oracle user-defined type name that the custom class maps to
TypeId	Inherited from System.Attribute

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleCustomTypeMappingAttribute Class](#)
- [OracleCustomTypeMappingAttribute Members](#)

UdtTypeName

This property specifies the Oracle user-defined type name that the custom class maps to.

Declaration

```
// C#  
public string UdtTypeName {get; set;}
```

Property Value

A string that represents an Oracle user-defined type name.

Remarks

UdtTypeName is case-sensitive. It is specified in the form of *schema_name.type_name*.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleCustomTypeMappingAttribute Class](#)
- [OracleCustomTypeMappingAttribute Members](#)

OracleCustomTypeMappingAttribute Methods

OracleCustomTypeMappingAttribute methods are listed in [Table 16-7](#).

Table 16-7 OracleCustomTypeMappingAttribute Methods

Method	Description
Equals	Inherited from System.Attribute
GetHashCode	Inherited from System.Attribute
GetType	Inherited from System.Attribute
IsDefaultAttribute	Inherited from System.Attribute
Match	Inherited from System.Attribute
ToString	Inherited from System.Attribute

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleCustomTypeMappingAttribute Class](#)
- [OracleCustomTypeMappingAttribute Members](#)

OracleObjectMappingAttribute Class

The OracleObjectMappingAttribute class marks custom class fields or properties with information that ODP.NET uses when a custom type represents an Oracle Object type.

Class Inheritance

System.Object
System.Attribute
System.OracleObjectMappingAttribute

Declaration

```
// C#  
[AttributeUsageAttribute(AttributeTargets.Field|AttributeTargets.Property,  
AllowMultiple=false, Inherited=true)]  
  
public sealed class OracleObjectMappingAttribute : Attribute
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Remarks

The OracleObjectMappingAttribute is specified on members of a custom type that represent an Oracle object type. This attribute must specify the name or zero-based index of the attribute in the Oracle object that the custom class field or property maps to. This also allows the custom type to declare field or property names which differ from the Oracle Object type.

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleObjectMappingAttribute Members](#)
- [OracleObjectMappingAttribute Constructors](#)
- [OracleObjectMappingAttribute Static Methods](#)
- [OracleObjectMappingAttribute Properties](#)
- [OracleObjectMappingAttribute Methods](#)

OracleObjectMappingAttribute Members

OracleObjectMappingAttribute members are listed in the following tables.

OracleObjectMappingAttribute Constructors

OracleObjectMappingAttribute constructors are listed in [Table 16-8](#).

Table 16-8 OracleObjectMappingAttribute Constructors

Constructor	Description
OracleObjectMappingAttribute Constructors	Instantiates a new instance of OracleObjectMappingAttribute class (Overloaded)

OracleObjectMappingAttribute Static Methods

OracleObjectMappingAttribute static methods are listed in [Table 16-9](#).

Table 16-9 OracleObjectMappingAttribute Static Methods

Method	Description
Equals	Inherited from System.Attribute
GetCustomAttribute	Inherited from System.Attribute
GetCustomAttributes	Inherited from System.Attribute
IsDefined	Inherited from System.Attribute
ReferenceEquals	Inherited from System.Attribute

OracleObjectMappingAttribute Properties

OracleObjectMappingAttribute properties are listed in [Table 16-10](#).

Table 16-10 OracleObjectMappingAttribute Properties

Property	Description
AttributeIndex	Specifies the index of the Oracle Object attribute that must be retrieved

Table 16-10 (Cont.) OracleObjectMappingAttribute Properties

Property	Description
AttributeName	Specifies the name of Oracle Object attribute that must be retrieved
TypeId	Inherited from System.Attribute

OracleObjectMappingAttribute Methods

OracleObjectMappingAttribute methods are listed in [Table 16-11](#).

Table 16-11 OracleObjectMappingAttribute Methods

Method	Description
Equals	Inherited from System.Attribute
GetHashCode	Inherited from System.Attribute
GetType	Inherited from System.Attribute
IsDefaultAttribute	Inherited from System.Attribute
Match	Inherited from System.Attribute
ToString	Inherited from System.Attribute

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleObjectMappingAttribute Class](#)

OracleObjectMappingAttribute Constructors

OracleObjectMappingAttribute constructors create new instances of the OracleObjectMappingAttribute class.

Overload List:

- [OracleObjectMappingAttribute\(string\)](#)
This constructor creates and initializes an OracleObjectMappingAttribute object with the specified Oracle Object attribute name.
- [OracleObjectMappingAttribute\(int\)](#)
This constructor creates and initializes an OracleObjectMappingAttribute with the specified Oracle Object attribute index.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleObjectMappingAttribute Class](#)
- [OracleObjectMappingAttribute Members](#)

OracleObjectMappingAttribute(string)

This constructor creates and initializes an `OracleObjectMappingAttribute` object with the specified Oracle Object attribute name.

Declaration

```
// C#  
public OracleObjectMappingAttribute(string attrName);
```

Parameters

- *attrName*

The name of the Oracle Object attribute to map to.

Remarks

The *attrName* parameter is case-sensitive.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleObjectMappingAttribute Class](#)
- [OracleObjectMappingAttribute Members](#)

OracleObjectMappingAttribute(int)

This constructor creates and initializes an `OracleObjectMappingAttribute` object with the specified Oracle Object attribute index.

Declaration

```
// C#  
public OracleObjectMappingAttribute(int attrIndex);
```

Parameters

- *attrIndex*

The zero-based index of the Oracle Object attribute to map to.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleObjectMappingAttribute Class](#)
- [OracleObjectMappingAttribute Members](#)

OracleObjectMappingAttribute Static Methods

OracleObjectMappingAttribute static methods are listed in [Table 16-12](#).

Table 16-12 OracleObjectMappingAttribute Static Method

Method	Description
Equals	Inherited from System.Attribute
GetCustomAttribute	Inherited from System.Attribute
GetCustomAttributes	Inherited from System.Attribute
IsDefined	Inherited from System.Attribute
ReferenceEquals	Inherited from System.Attribute

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleObjectMappingAttribute Class](#)
- [OracleObjectMappingAttribute Members](#)

OracleObjectMappingAttribute Properties

OracleObjectMappingAttribute properties are listed in [Table 16-13](#).

Table 16-13 OracleObjectMappingAttribute Properties

Property	Description
AttributeIndex	Specifies the index of the Oracle Object attribute that must be retrieved
AttributeName	Specifies the name of the Oracle Object attribute that must be retrieved
TypeId	Inherited from System.Attribute

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleObjectMappingAttribute Class](#)
- [OracleObjectMappingAttribute Members](#)

AttributeIndex

This property specifies the index of the Oracle Object attribute that must be retrieved.

Declaration

```
// C#  
public int AttributeIndex {get;}
```

Property Value

The zero-based index of an Oracle Object type attribute.

Remarks

The AttributeIndex property specifies the index of the Oracle Object type attribute that the custom class field or property maps to. This allows the custom class to declare fields or property names that differ from the Oracle object.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleObjectMappingAttribute Class](#)
- [OracleObjectMappingAttribute Members](#)

AttributeName

This property specifies the name of the Oracle Object attribute that must be retrieved.

Declaration

```
// C#  
public string AttributeName {get;}
```

Property Value

The name of an attribute of an Oracle Object type.

Remarks

The AttributeName property specifies name of the attribute in the Oracle Object type that the custom class field or property maps to. This allows the custom class to declare field or property names that differ from the Oracle object.

The specified attribute name is case-sensitive.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleObjectMappingAttribute Class](#)
- [OracleObjectMappingAttribute Members](#)

OracleObjectMappingAttribute Methods

OracleObjectMappingAttribute methods are listed in [Table 16-14](#).

Table 16-14 OracleObjectMappingAttribute Methods

Method	Description
Equals	Inherited from System.Attribute
GetHashCode	Inherited from System.Attribute
GetType	Inherited from System.Attribute
IsDefaultAttribute	Inherited from System.Attribute
Match	Inherited from System.Attribute
ToString	Inherited from System.Attribute

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleObjectMappingAttribute Class](#)
- [OracleObjectMappingAttribute Members](#)

OracleArrayMappingAttribute Class

The OracleArrayMappingAttribute class is required to mark a custom class field or property with information that ODP.NET uses when a custom type represents an Oracle Collection type.

Class Inheritance

System.Object
 System.Attribute
 System.OracleArrayMappingAttribute

Declaration

```
[AttributeUsageAttribute(AttributeTargets.Field|AttributeTargets.Property, AllowMultiple=false,
Inherited=true)]

// C#
public sealed class OracleArrayMappingAttribute : Attribute
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Remarks

An OracleArrayMappingAttribute object must be specified when a custom type represents an Oracle Collection. This attribute is applied only to the custom class member that stores the collection elements.

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleArrayMappingAttribute Members](#)
- [OracleArrayMappingAttribute Constructors](#)
- [OracleArrayMappingAttribute Static Methods](#)
- [OracleArrayMappingAttribute Properties](#)
- [OracleArrayMappingAttribute Methods](#)

OracleArrayMappingAttribute Members

OracleArrayMappingAttribute members are listed in the following tables.

OracleArrayMappingAttribute Constructors

OracleArrayMappingAttribute constructors are listed in [Table 16-15](#).

Table 16-15 OracleArrayMappingAttribute Constructors

Constructor	Description
OracleArrayMappingAttribute Constructors	Instantiates a new instance of OracleArrayMappingAttribute class (Overloaded)

OracleArrayMappingAttribute Static Methods

OracleArrayMappingAttribute static methods are listed in [Table 16-16](#).

Table 16-16 OracleArrayMappingAttribute Static Methods

Method	Description
Equals	Inherited from System.Attribute
GetCustomAttribute	Inherited from System.Attribute
GetCustomAttributes	Inherited from System.Attribute
IsDefined	Inherited from System.Attribute
ReferenceEquals	Inherited from System.Attribute

OracleArrayMappingAttribute Properties

OracleArrayMappingAttribute properties are listed in [Table 16-17](#).

Table 16-17 OracleArrayMappingAttribute Properties

Property	Description
TypeId	Inherited from System.Attribute

OracleArrayMappingAttribute Methods

OracleArrayMappingAttribute methods are listed in [Table 16-18](#).

Table 16-18 OracleArrayMappingAttribute Methods

Method	Description
Equals	Inherited from System.Attribute
GetHashCode	Inherited from System.Attribute
GetType	Inherited from System.Attribute
IsDefaultAttribute	Inherited from System.Attribute
Match	Inherited from System.Attribute
ToString	Inherited from System.Attribute

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleArrayMappingAttribute Class](#)

OracleArrayMappingAttribute Constructors

OracleArrayMappingAttribute constructors create new instances of the OracleArrayMappingAttribute class.

Overload List:

- [OracleArrayMappingAttribute\(\)](#)
This constructor creates and initializes an OracleArrayMappingAttribute object.

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleArrayMappingAttribute Class](#)
- [OracleArrayMappingAttribute Members](#)

OracleArrayMappingAttribute()

This constructor creates and initializes an OracleArrayMappingAttribute object.

Declaration

```
// C#
public OracleArrayMappingAttribute();
```

Remarks

An OracleArrayMappingAttribute object must be applied when a custom class represents an Oracle Collection type, to specify the custom class field or property that stores the collection elements.

The OracleArrayMappingAttribute can be applied to only one field or property in the custom class.

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleArrayMappingAttribute Class](#)
- [OracleArrayMappingAttribute Members](#)

OracleArrayMappingAttribute Static Methods

OracleArrayMappingAttribute static methods are listed in [Table 16-19](#).

Table 16-19 OracleArrayMappingAttribute Static Methods

Method	Description
Equals	Inherited from System.Attribute
GetCustomAttribute	Inherited from System.Attribute
GetCustomAttributes	Inherited from System.Attribute

Table 16-19 (Cont.) OracleArrayMappingAttribute Static Methods

Method	Description
IsDefined	Inherited from System.Attribute
ReferenceEquals	Inherited from System.Attribute

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleArrayMappingAttribute Class](#)
- [OracleArrayMappingAttribute Members](#)

OracleArrayMappingAttribute Properties

OracleArrayMappingAttribute properties are listed in [Table 16-20](#).

Table 16-20 OracleArrayMappingAttribute Properties

Property	Description
TypeId	Inherited from System.Attribute

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleArrayMappingAttribute Class](#)
- [OracleArrayMappingAttribute Members](#)

OracleArrayMappingAttribute Methods

OracleArrayMappingAttribute methods are listed in [Table 16-21](#).

Table 16-21 OracleArrayMappingAttribute Methods

Method	Description
Equals	Inherited from System.Attribute
GetHashCode	Inherited from System.Attribute
GetType	Inherited from System.Attribute
IsDefaultAttribute	Inherited from System.Attribute
Match	Inherited from System.Attribute
ToString	Inherited from System.Attribute

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleArrayMappingAttribute Class](#)
- [OracleArrayMappingAttribute Members](#)

IOracleCustomType Interface

IOracleCustomType is an interface for converting between a Custom Type and an Oracle Object or Collection Type.

Declaration

```
// C#
public interface IOracleCustomType
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [IOracleCustomType Members](#)
- [IOracleCustomType Interface Methods](#)

IOracleCustomType Members

IOracleCustomType members are listed in the following tables.

IOracleCustomType Interface Methods

IOracleCustomType interface methods are listed in [Table 16-22](#).

Table 16-22 IOracleCustomType Interface Methods

Interface Method	Description
FromCustomObject(OracleConnection, IntPtr)	Returns the values that set the Oracle Object attributes <i>Available in ODP.NET, Unmanaged Driver only</i>
FromCustomObject(OracleConnection, object)	Returns the values that set the Oracle Object attributes <i>Available in ODP.NET, Managed Driver and ODP.NET Core only</i>
ToCustomObject(OracleConnection, IntPtr)	Provides the Oracle Object with the attribute values to set on the custom type <i>Available in ODP.NET, Unmanaged Driver only</i>
ToCustomObject(OracleConnection, object)	Provides the Oracle Object with the attribute values to set on the custom type <i>Available in ODP.NET, Managed Driver and ODP.NET Core only</i>

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [IOracleCustomType Interface](#)

IOracleCustomType Interface Methods

IOracleCustomType Interface methods are listed in [Table 16-23](#).

Table 16-23 IOracleCustomType Interface Methods: Overload list

Interface Method	Description
FromCustomObject(OracleConnection, IntPtr)	Returns the values that set the Oracle Object attributes <i>Available in ODP.NET, Unmanaged Driver only</i>
FromCustomObject(OracleConnection, object)	Returns the values that set the Oracle Object attributes <i>Available in ODP.NET, Managed Driver and ODP.NET Core only</i>
ToCustomObject(OracleConnection, IntPtr)	Provides the Oracle Object with the attribute values to set on the custom type <i>Available in ODP.NET, Unmanaged Driver only</i>
ToCustomObject(OracleConnection, object)	Provides the Oracle Object with the attribute values to set on the custom type <i>Available in ODP.NET, Managed Driver and ODP.NET Core only</i>

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [IOracleCustomType Interface](#)
- [IOracleCustomType Members](#)

FromCustomObject(OracleConnection, IntPtr)

This interface method creates an Oracle Object or Collection by setting the attribute or element values respectively on the specified Oracle UDT.

Declaration

```
// C#  
void FromCustomObject(OracleConnection con, IntPtr pUdt);
```

Parameters

- *con*
An OracleConnection instance.
- *pUdt*
An opaque pointer to the Oracle Object or Collection to be created.

Remarks

The FromCustomObject method is used to build an Oracle Object or Collection from a custom object by setting attribute or element values respectively through the OracleUdt.SetValue method.

The OracleUdt.SetValue method is invoked as follows:

- Oracle Object Type
For a custom type that represents an Oracle Object Type, the OracleUdt.SetValue method must be invoked for each non-NULL attribute value that needs to be set.
- Oracle Collection Type
For a custom type that represents an Oracle Collection Type, a single call to OracleUdt.SetValue method specifies the collection element values.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [IOracleCustomType Interface](#)
- [IOracleCustomType Members](#)

FromCustomObject(OracleConnection, object)

This interface method creates an Oracle Object or Collection by setting the attribute or element values respectively on the specified Oracle UDT.

Declaration

```
// C#  
void FromCustomObject(OracleConnection con, object pUdt);
```

Parameters

- *con*
An OracleConnection instance.
- *pUdt*
An object of Oracle Object or Collection to be created.

Remarks

The FromCustomObject method is used to build an Oracle Object or Collection from a custom object by setting attribute or element values respectively through the OracleUdt.SetValue method.

The OracleUdt.SetValue method is invoked as follows:

- Oracle Object Type
For a custom type that represents an Oracle Object Type, the OracleUdt.SetValue method must be invoked for each non-NULL attribute value that needs to be set.
- Oracle Collection Type
For a custom type that represents an Oracle Collection Type, a single call to OracleUdt.SetValue method specifies the collection element values.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [IOracleCustomType Interface](#)
- [IOracleCustomType Members](#)

ToCustomObject(OracleConnection, IntPtr)

This interface initializes a custom object using the specified Oracle UDT.

Declaration

```
// C#  
void ToCustomObject (OracleConnection con, IntPtr pUdt);
```

Parameters

- *con*
An OracleConnection instance.
- *pUdt*
An opaque pointer to the Oracle UDT.

Remarks

The ToCustomObject method is used to initialize a custom object from the specified Oracle Object or Collection by retrieving attribute or element values respectively through the OracleUdt.GetValue method.

The `OracleUdt.GetValue` method is invoked as follows:

- **Oracle Object Type**
For a custom type that represents an Oracle Object Type, the `OracleUdt.GetValue` method must be invoked for each attribute value to be retrieved.
- **For a custom type that represents an Oracle Collection Type**, a single call to `OracleUdt.GetValue` method retrieves the collection element values.

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [IOracleCustomType Interface](#)
- [IOracleCustomType Members](#)

ToCustomObject(OracleConnection, object)

This interface initializes a custom object using the specified Oracle UDT.

Declaration

```
// C#  
void ToCustomObject (OracleConnection con, object pUdt);
```

Parameters

- *con*
An `OracleConnection` instance.
- *pUdt*
An object of the Oracle UDT.

Remarks

The `ToCustomObject` method is used to initialize a custom object from the specified Oracle Object or Collection by retrieving attribute or element values respectively through the `OracleUdt.GetValue` method.

The `OracleUdt.GetValue` method is invoked as follows:

- **Oracle Object Type**
For a custom type that represents an Oracle Object Type, the `OracleUdt.GetValue` method must be invoked for each attribute value to be retrieved.
- **For a custom type that represents an Oracle Collection Type**, a single call to `OracleUdt.GetValue` method retrieves the collection element values.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [IOracleCustomType Interface](#)
- [IOracleCustomType Members](#)

IOracleCustomTypeFactory Interface

The IOracleCustomTypeFactory interface is used by ODP.NET to create custom objects that represent Oracle Objects or Collections.

Declaration

```
// C#
public interface IOracleCustomTypeFactory
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [IOracleCustomTypeFactory Members](#)
- [IOracleCustomTypeFactory Interface Methods](#)

IOracleCustomTypeFactory Members

IOracleCustomTypeFactory members are listed in the following tables.

IOracleCustomTypeFactory Interface Methods

IOracleCustomTypeFactory interface methods are listed in [Table 16-24](#).

Table 16-24 IOracleCustomTypeFactory Interface Methods

Public Method	Description
CreateObject	Returns a new custom object to represent an Oracle Object or Collection

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [IOracleCustomTypeFactory Interface](#)

IOracleCustomTypeFactory Interface Methods

IOracleCustomTypeFactory Interface methods are listed in [Table 16-25](#).

Table 16-25 IOracleCustomTypeFactory Interface Methods

Public Method	Description
CreateObject	Returns a new custom object to represent an Oracle Object or Collection

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [IOracleCustomTypeFactory Interface](#)
- [IOracleCustomTypeFactory Members](#)

CreateObject

This interface method returns a new custom object to represent an Oracle Object or Collection.

Declaration

```
// C#  
IOracleCustomType CreateObject();
```

Return Value

An IOracleCustomType object.

Remarks

The CreateObject method is used to create a new instance of a custom object to represent an Oracle Object or Collection.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [IOracleCustomTypeFactory Interface](#)
- [IOracleCustomTypeFactory Members](#)

IOracleArrayTypeFactory Interface

The IOracleArrayTypeFactory interface is used by ODP.NET to create arrays that represent Oracle Collections.

Declaration

```
// C#
public interface IOracleArrayTypeFactory
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [IOracleArrayTypeFactory Members](#)
- [IOracleArrayTypeFactory Interface Methods](#)

IOracleArrayTypeFactory Members

IOracleArrayTypeFactory members are listed in the following tables.

IOracleArrayTypeFactory Interface Methods

IOracleArrayTypeFactory interface methods are listed in [Table 16-26](#).

Table 16-26 IOracleArrayTypeFactory Interface Methods

Public Method	Description
CreateArray	Returns a new array of the specified length to store Oracle Collection elements
CreateStatusArray	Returns a newly allocated OracleUdtStatus array of the specified length that will be used to store the null status of the collection elements

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [IOracleArrayTypeFactory Interface](#)

IOracleArrayTypeFactory Interface Methods

IOracleArrayTypeFactory Interface methods are listed in [Table 16-27](#).

Table 16-27 IOracleArrayTypeFactory Interface Methods

Public Method	Description
CreateArray	Returns a new array of the specified length to store Oracle Collection elements
CreateStatusArray	Returns a newly allocated OracleUdtStatus array of the specified length that will be used to store the null status of the collection elements

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [IOracleArrayTypeFactory Interface](#)
- [IOracleArrayTypeFactory Members](#)

CreateArray

This interface method returns a new array of the specified length to store Oracle Collection elements.

Declaration

```
// C#  
Array CreateArray(int numElems);
```

Parameters

- *numElems*

The number of collection elements to be returned.

Return Value

A `System.Array` object.

Remarks

An Oracle Collection Type may be represented in either of the following ways:

- As an array of the appropriate type. The type must be able to represent a collection element.
- As a Custom Type that contains an array of the appropriate type.

In both cases, the `CreateArray` method creates an array of the specified length to store the collection elements.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [IOracleArrayTypeFactory Interface](#)
- [IOracleArrayTypeFactory Members](#)

CreateStatusArray

This method returns a newly allocated `OracleUdtStatus` array of the specified length that will be used to store the null status of the collection elements.

Declaration

```
// C#  
Array CreateStatusArray(int numElems);
```

Parameters

- *numElems*

The number of collection elements to be returned.

Return Value

A multi-dimensional `OracleUdtStatus` array as a `System.Array`.

Remarks

An Oracle Collection Type can be represented in the following ways:

- As an array of the appropriate type. The type must be able to represent a collection element.
- As a Custom Type that contains an array of the appropriate type.

In both cases, the `CreateStatusArray` method creates an `OracleUdtStatus` array of the specified length that stores the null status of the collection elements.

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [IOracleArrayTypeFactory Interface](#)
- [IOracleArrayTypeFactory Members](#)
- ["OracleUdtFetchOption Enumeration"](#)

OracleUdt Class

The `OracleUdt` class defines static methods that are used when converting between Custom Types and Oracle UDTs and vice-versa.

Class Inheritance

System.Object
 System.OracleUdt

Declaration

public sealed class OracleUdt

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleUdt Members](#)
- [OracleUDT Static Methods](#)

OracleUdt Members

OracleUdt static methods are listed in [Table 16-28](#).

Table 16-28 OracleUdt Static Methods

Static Method	Description
Equals	Inherited from System.Object
GetValue	Gets the attributes or elements from the specified Oracle UDT (Overloaded)
IsDBNull	Indicates whether or not the specified attribute being retrieved is NULL (Overloaded)
SetValue	Sets the attributes or elements on the specified Oracle UDT (Overloaded)

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleUdt Class](#)

OracleUDT Static Methods

OracleUDT methods are listed in [Table 16-29](#).

Table 16-29 OracleUdt Static Methods

Static Method	Description
Equals	Inherited from System.Object
GetValue	Gets the attributes or elements from the specified Oracle UDT (Overloaded)
IsDBNull	Indicates whether or not the specified attribute being retrieved is NULL (Overloaded)
SetValue	Sets the attributes or elements on the specified Oracle UDT (Overloaded)

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleUdt Class](#)
- [OracleUdt Members](#)

GetValue

GetValue methods get the attributes or elements from the specified Oracle UDT.

Overload List for ODP.NET, Unmanaged Driver:

- [GetValue\(OracleConnection, IntPtr, string\)](#)
This method gets the attributes or elements from the specified Oracle UDT, using the specified attribute name.
- [GetValue\(OracleConnection, IntPtr, int\)](#)
This method gets the attribute or elements from the specified Oracle UDT, using the specified index.
- [GetValue\(OracleConnection, IntPtr, string, out object\)](#)
This method returns either the elements of the specified collection attribute of the specified Oracle Object or the elements of the specified Oracle Collection.
- [GetValue\(OracleConnection, IntPtr, int, out object\)](#)
This method returns either the elements of the specified collection attribute of the specified Oracle Object or the elements of the specified Oracle Collection.

Overload List for ODP.NET, Managed Driver and ODP.NET Core:

- [GetValue\(OracleConnection, object, string\)](#)
This method gets the attributes or elements from the specified Oracle UDT, using the specified attribute name.
- [GetValue\(OracleConnection, object, int\)](#)
This method gets the attribute or elements from the specified Oracle UDT, using the specified index.
- [GetValue\(OracleConnection, object, string, out object\)](#)
This method returns either the elements of the specified collection attribute of the specified Oracle Object or the elements of the specified Oracle Collection.
- [GetValue\(OracleConnection, object, int, out object\)](#)
This method returns either the elements of the specified collection attribute of the specified Oracle Object or the elements of the specified Oracle Collection.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleUdt Class](#)
- ["OracleUdt Members"](#)

GetValue(OracleConnection, IntPtr, string)

This method gets the attributes or elements from the specified Oracle UDT, using the specified attribute name.

Declaration

```
public static object GetValue(OracleConnection con, IntPtr pUdt, string attrName);
```

Parameters

- *con*
An OracleConnection instance.
- *pUdt*
A pointer to an Oracle UDT.
- *attrName*
The case-sensitive name of the attribute to be retrieved. Null is specified for retrieving collection elements from a Custom Type that represents an Oracle Collection.

Return Value

An object representing the returned attribute or collection elements.

Exceptions

ArgumentException - The specified name is not a valid attribute name.

Remarks

The `IOracleCustomType.ToCustomObject` method invokes `OracleUdt.GetValue` method passing it the *con* and *pUdt* parameters. The `OracleUdt.GetValue` method returns these types of object:

- Oracle Object Type
For a Custom Type that represents an Oracle Object Type, the type returned for a specified attribute name is the type of the member in the custom class or struct that is mapped to the attribute using the `OracleObjectMappingAttribute` object.
- Oracle Collection Type
For a Custom Type that represents an Oracle Collection Type, the type returned is the type of the member in the custom class or struct to which the `OracleArrayMappingAttribute` object is applied.

In the case of NULL attribute values, the appropriate null representation of the type is returned. For example, for attributes that are represented as Custom Types and Provider Specific Types, the static `Null` property of the type is returned. For attributes that are represented as Nullable

types, for example, `System.String` and `System.Array` Types, null is returned, and for all other remaining built-in types such as `Int32` and `DateTime` `DBNull.Value` is returned.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleUdt Class](#)
- [OracleUdt Members](#)
- ["OracleUdtFetchOption Enumeration"](#)

GetValue(OracleConnection, IntPtr, int)

This method gets the attribute or elements from the specified Oracle UDT, using the specified index.

Declaration

```
// C#  
public static object GetValue(OracleConnection con, IntPtr pUdt, int attrIndex,);
```

Parameters

- *con*
An `OracleConnection` instance.
- *pUdt*
A pointer to an Oracle UDT.
- *attrIndex*
The zero-based index of the attribute to be retrieved. For retrieving collection elements from a Custom Type that represents an Oracle Collection, zero must be specified.

Return Value

An object representing the returned attribute or collection elements.

Exceptions

`ArgumentOutOfRangeException` - The specified index is not a valid attribute index.

Remarks

The `IOracleCustomType.ToCustomObject` method invokes `OracleUdt.GetValue` method passing it the *con* and *pUdt* parameters. The `OracleUdt.GetValue` method returns these types of object:

- Oracle Object Type
For a Custom Type that represents an Oracle Object Type, the type returned for a specified attribute index is the type of the member in the custom class or struct that is mapped to the attribute using the `OracleObjectMappingAttribute` object.
- Oracle Collection Type

For a Custom Type that represents an Oracle Collection Type, the type returned is the type of the member in the custom class or struct to which the `OracleArrayMappingAttribute` object is applied.

In the case of NULL attribute values, the appropriate null representation of the type is returned. For example, for attributes that are represented as Custom Types and Provider Specific Types, the static `Null` property of the type is returned. For attributes that are represented as Nullable types, for example, `System.String` and `System.Array` Types, null is returned, and for all other remaining built-in types such as `Int32` and `DateTime` `DBNull.Value` is returned.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleUdt Class](#)
- [OracleUdt Members](#)
- ["OracleUdtFetchOption Enumeration"](#)

GetValue(OracleConnection, IntPtr, string, out object)

This method returns either the elements of the specified collection attribute of the specified Oracle Object or the elements of the specified Oracle Collection.

Declaration

```
// C#
public static object GetValue(OracleConnection con, IntPtr pUdt, string attrName,
    out object statusArray);
```

Parameters

- *con*
An `OracleConnection` instance.
- *pUdt*
An opaque pointer to an Oracle UDT.
- *attrName*
The case-sensitive name of the attribute to be retrieved. Null must specified for retrieving collection elements from a Custom Type that represents an Oracle Collection.
- *statusArray*
The `OracleUdtStatus` array which returns the null status for the retrieved collection elements.

Return Value

An object representing the returned attribute or collection elements.

Exceptions

`ArgumentException` - The specified name is not a valid attribute name.

Remarks

The `IOracleCustomType.ToCustomObject` method invokes `OracleUdt.GetValue` method passing it the *con* and *pUdt* parameters. The `OracleUdt.GetValue` method returns these types of object:

- Oracle Object Type

For a Custom Type that represents an Oracle Object Type, the type returned for a specified attribute name is the type of the member in the custom class or struct that is mapped to the attribute using the `OracleObjectMappingAttribute` object.

- Oracle Collection Type

For a Custom Type that represents an Oracle Collection Type, the type returned is the type of the member in the custom class or struct to which the `OracleArrayMappingAttribute` object is applied.

In the case of NULL attribute values, the appropriate null representation of the type is returned. For example, for attributes that are represented as Custom Types and Provider Specific Types, the static `Null` property of the type is returned. For attributes that are represented as Nullable types, for example, `System.String` and `System.Array` Types, null is returned, and for all other remaining built-in types such as `Int32` and `DateTime` `DBNull.Value` is returned.

If the collection being returned is not NULL, the output *statusArray* parameter is populated with the null status for each of the collection elements.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleUdt Class](#)
- [OracleUdt Members](#)
- ["OracleUdtFetchOption Enumeration"](#)

GetValue(OracleConnection, IntPtr, int, out object)

This method returns either the elements of the specified collection attribute of the specified Oracle Object or the elements of the specified Oracle Collection.

Declaration

```
// C#
public static object GetValue(OracleConnection con, IntPtr pUdt, int attrIndex,
    out object statusArray);
```

Parameters

- *con*
An `OracleConnection` instance.
- *pUdt*
An opaque pointer to an Oracle UDT.
- *attrIndex*

The zero-based index of the attribute to be retrieved. For retrieving collection elements from a Custom Type that represents an Oracle Collection, 0 is specified.

- *statusArray*

The `OracleUdtStatus` array which returns the null status for the retrieved collection elements.

Return Value

An object representing the returned attribute or collection elements.

Exceptions

`ArgumentOutOfRangeException` - The specified index is not a valid attribute index.

Remarks

The `IOracleCustomType.ToCustomObject` method invokes `OracleUdt.GetValue` method passing it the *con* and *pUdt* parameters. The `OracleUdt.GetValue` method returns these types of object:

- Oracle Object Type

For a Custom Type that represents an Oracle Object Type, the type returned for a specified attribute index is the type of the member in the custom class or struct that is mapped to the attribute using the `OracleObjectMappingAttribute` object.

- Oracle Collection Type

For a Custom Type that represents an Oracle Collection Type, the type returned is the type of the member in the custom class or struct to which the `OracleArrayMappingAttribute` object is applied.

In the case of NULL attribute values, the appropriate null representation of the type is returned. For example, for attributes that are represented as Custom Types and Provider Specific Types, the static `Null` property of the type is returned. For attributes that are represented as Nullable types, for example, `System.String` and `System.Array` Types, null is returned, and for all other remaining built-in types such as `Int32` and `DateTime` `DBNull.Value` is returned.

If the collection being returned is not NULL, the output *statusArray* parameter is populated with the null status for each of the collection elements.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleUdt Class](#)
- [OracleUdt Members](#)
- ["OracleUdtFetchOption Enumeration"](#)

GetValue(OracleConnection, object, string)

This method gets the attributes or elements from the specified Oracle UDT, using the specified attribute name.

Declaration

```
public static object GetValue(OracleConnection con, object Udt, string attrName);
```

Parameters

- *con*
An OracleConnection instance.
- *Udt*
An object of an Oracle UDT.
- *attrName*
The case-sensitive name of the attribute to be retrieved. Null is specified for retrieving collection elements from a Custom Type that represents an Oracle Collection.

Return Value

An object representing the returned attribute or collection elements.

Exceptions

ArgumentException - The specified name is not a valid attribute name.

Remarks

The `IOracleCustomType.ToCustomObject` method invokes `OracleUdt.GetValue` method passing it the *con* and *Udt* parameters. The `OracleUdt.GetValue` method returns these types of object:

- Oracle Object Type
For a Custom Type that represents an Oracle Object Type, the type returned for a specified attribute name is the type of the member in the custom class or struct that is mapped to the attribute using the `OracleObjectMappingAttribute` object.
- Oracle Collection Type
For a Custom Type that represents an Oracle Collection Type, the type returned is the type of the member in the custom class or struct to which the `OracleArrayMappingAttribute` object is applied.

In the case of NULL attribute values, the appropriate null representation of the type is returned. For example, for attributes that are represented as Custom Types and Provider Specific Types, the static `Null` property of the type is returned. For attributes that are represented as Nullable types, for example, `System.String` and `System.Array` Types, null is returned, and for all other remaining built-in types such as `Int32` and `DateTime` `DBNull.Value` is returned.

If the type of the member is mapped to a .NET Decimal, and `SuppressGetDecimalInvalidCastException` is set to true on `OracleConfiguration`, `OracleConnection`, `OracleDataAdapter`, or `OracleDataReader` associated with the UDT, then it returns a rounded-off 28 or 29 precision Oracle NUMBER value that can be represented as a .NET Decimal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleUdt Class](#)
- [OracleUdt Members](#)
- ["OracleUdtFetchOption Enumeration"](#)

GetValue(OracleConnection, object, int)

This method gets the attribute or elements from the specified Oracle UDT, using the specified index.

Declaration

```
// C#  
public static object GetValue(OracleConnection con, object Udt, int attrIndex,);
```

Parameters

- *con*
An OracleConnection instance.
- *Udt*
An Oracle UDT object.
- *attrIndex*
The zero-based index of the attribute to be retrieved. For retrieving collection elements from a Custom Type that represents an Oracle Collection, zero must be specified.

Return Value

An object representing the returned attribute or collection elements.

Exceptions

ArgumentOutOfRangeException - The specified index is not a valid attribute index.

Remarks

The `IOracleCustomType.ToCustomObject` method invokes `OracleUdt.GetValue` method passing it the *con* and *Udt* parameters. The `OracleUdt.GetValue` method returns these types of object:

- Oracle Object Type
For a Custom Type that represents an Oracle Object Type, the type returned for a specified attribute index is the type of the member in the custom class or struct that is mapped to the attribute using the `OracleObjectMappingAttribute` object.
- Oracle Collection Type
For a Custom Type that represents an Oracle Collection Type, the type returned is the type of the member in the custom class or struct to which the `OracleArrayMappingAttribute` object is applied.

In the case of NULL attribute values, the appropriate null representation of the type is returned. For example, for attributes that are represented as Custom Types and Provider Specific Types, the static Null property of the type is returned. For attributes that are represented as Nullable types, for example, System.String and System.Array Types, null is returned, and for all other remaining built-in types such as Int32 and DateTime DBNull.Value is returned.

If the type of the member is mapped to a .NET Decimal, and SuppressGetDecimalInvalidCastException is set to true on OracleConfiguration, OracleConnection, OracleDataAdapter, or OracleDataReader associated with the UDT, then it returns a rounded-off 28 or 29 precision Oracle NUMBER value that can be represented as a .NET Decimal.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleUdt Class](#)
- [OracleUdt Members](#)
- ["OracleUdtFetchOption Enumeration"](#)

GetValue(OracleConnection, object, string, out object)

This method returns either the elements of the specified collection attribute of the specified Oracle Object or the elements of the specified Oracle Collection.

Declaration

```
// C#  
public static object GetValue(OracleConnection con, object Udt, string attrName,  
    out object statusArray);
```

Parameters

- *con*
An OracleConnection instance.
- *pUdt*
An Oracle UDT object.
- *attrName*
The case-sensitive name of the attribute to be retrieved. Null must specified for retrieving collection elements from a Custom Type that represents an Oracle Collection.
- *statusArray*
The OracleUdtStatus array which returns the null status for the retrieved collection elements.

Return Value

An object representing the returned attribute or collection elements.

Exceptions

ArgumentException - The specified name is not a valid attribute name.

Remarks

The `IOracleCustomType.ToCustomObject` method invokes `OracleUdt.GetValue` method passing it the `con` and `Udt` parameters. The `OracleUdt.GetValue` method returns these types of object:

- Oracle Object Type

For a Custom Type that represents an Oracle Object Type, the type returned for a specified attribute name is the type of the member in the custom class or struct that is mapped to the attribute using the `OracleObjectMappingAttribute` object.

- Oracle Collection Type

For a Custom Type that represents an Oracle Collection Type, the type returned is the type of the member in the custom class or struct to which the `OracleArrayMappingAttribute` object is applied.

In the case of NULL attribute values, the appropriate null representation of the type is returned. For example, for attributes that are represented as Custom Types and Provider Specific Types, the static `Null` property of the type is returned. For attributes that are represented as Nullable types, for example, `System.String` and `System.Array` Types, null is returned, and for all other remaining built-in types such as `Int32` and `DateTime` `DBNull.Value` is returned.

If the collection being returned is not NULL, the output `statusArray` parameter is populated with the null status for each of the collection elements.

If the type of the member is mapped to a .NET Decimal, and `SuppressGetDecimalInvalidCastException` is set to true on `OracleConfiguration`, `OracleConnection`, `OracleDataAdapter`, or `OracleDataReader` associated with the UDT, then it returns a rounded-off 28 or 29 precision Oracle NUMBER value that can be represented as a .NET Decimal.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleUdt Class](#)
- [OracleUdt Members](#)
- ["OracleUdtFetchOption Enumeration"](#)

GetValue(OracleConnection, object, int, out object)

This method returns either the elements of the specified collection attribute of the specified Oracle Object or the elements of the specified Oracle Collection.

Declaration

```
// C#  
public static object GetValue(OracleConnection con, object Udt, int attrIndex,  
    out object statusArray);
```

Parameters

- `con`

An `OracleConnection` instance.

- *Udt*

An Oracle UDT object.

- *attrIndex*

The zero-based index of the attribute to be retrieved. For retrieving collection elements from a Custom Type that represents an Oracle Collection, 0 is specified.

- *statusArray*

The `OracleUdtStatus` array which returns the null status for the retrieved collection elements.

Return Value

An object representing the returned attribute or collection elements.

Exceptions

`ArgumentOutOfRangeException` - The specified index is not a valid attribute index.

Remarks

The `IOracleCustomType.ToCustomObject` method invokes `OracleUdt.GetValue` method passing it the *con* and *Udt* parameters. The `OracleUdt.GetValue` method returns these types of object:

- Oracle Object Type

For a Custom Type that represents an Oracle Object Type, the type returned for a specified attribute index is the type of the member in the custom class or struct that is mapped to the attribute using the `OracleObjectMappingAttribute` object.

- Oracle Collection Type

For a Custom Type that represents an Oracle Collection Type, the type returned is the type of the member in the custom class or struct to which the `OracleArrayMappingAttribute` object is applied.

In the case of NULL attribute values, the appropriate null representation of the type is returned. For example, for attributes that are represented as Custom Types and Provider Specific Types, the static `Null` property of the type is returned. For attributes that are represented as Nullable types, for example, `System.String` and `System.Array` Types, null is returned, and for all other remaining built-in types such as `Int32` and `DateTime` `DBNull.Value` is returned.

If the collection being returned is not NULL, the output *statusArray* parameter is populated with the null status for each of the collection elements.

If the type of the member is mapped to a .NET Decimal, and `SuppressGetDecimalInvalidCastException` is set to true on `OracleConfiguration`, `OracleConnection`, `OracleDataAdapter`, or `OracleDataReader` associated with the UDT, then it returns a rounded-off 28 or 29 precision Oracle NUMBER value that can be represented as a .NET Decimal.

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleUdt Class](#)
- [OracleUdt Members](#)
- "[OracleUdtFetchOption Enumeration](#)"

IsDBNull

IsDBNull methods indicate whether or not the specified attribute being retrieved is NULL.

Overload List for ODP.NET, Unmanaged Driver:

- [IsDBNull\(OracleConnection, IntPtr, string\)](#)
This method indicates whether or not the attribute being retrieved, specified by OracleConnection, pointer, and attribute name, is NULL.
- [IsDBNull\(OracleConnection, IntPtr, int\)](#)
This method indicates whether or not the attribute being retrieved, specified by OracleConnection, pointer, and attribute index, is NULL.

Overload List for ODP.NET, Managed Driver and ODP.NET Core :

- [IsDBNull\(OracleConnection, object, string\)](#)
This method indicates whether or not the attribute being retrieved, specified by OracleConnection, UDT, and attribute name, is NULL.
- [IsDBNull\(OracleConnection, object, int\)](#)
This method indicates whether or not the attribute being retrieved, specified by OracleConnection, UDT, and attribute index, is NULL.

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleUdt Class](#)
- [OracleUdt Members](#)
- "[OracleUdtFetchOption Enumeration](#)"

IsDBNull(OracleConnection, IntPtr, string)

This method indicates whether or not the attribute being retrieved, specified by OracleConnection, pointer, and attribute name, is NULL.

Declaration

```
// C#  
public static bool IsDBNull(OracleConnection con, IntPtr pUdt, string attrName);
```

Parameters

- *con*
An `OracleConnection` instance.
- *pUdt*
A pointer to an Oracle UDT.
- *attrName*
The case-sensitive name of the attribute.

Return Value

True if the specified attribute is NULL; otherwise, false.

Exceptions

`ArgumentException` - The specified name is not a valid attribute name.

Remarks

This method is invoked from the `IOracleCustomType.ToCustomObject` method. The *con* and *pUdt* parameter is passed from the `IOracleCustomType.ToCustomObject` method to the `OracleUdt.IsDBNull` method. The *attrName* parameter is case-sensitive.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleUdt Class](#)
- [OracleUdt Members](#)

IsDBNull(OracleConnection, IntPtr, int)

This method indicates whether or not the attribute being retrieved, specified by `OracleConnection`, pointer, and attribute index, is NULL.

Declaration

```
// C#  
public static bool IsDBNull(OracleConnection con, IntPtr pUdt, int attrIndex);
```

Parameters

- *con*
An `OracleConnection` instance.

- *pUdt*
An opaque pointer to an Oracle UDT.
- *attrIndex*
The zero-based index of the attribute.

Return Value

True if the specified attribute is NULL; otherwise, false.

Exceptions

`ArgumentOutOfRangeException` - The specified index is not a valid attribute index

Remarks

This method is invoked from the `IOracleCustomType.ToCustomObject` method. The *con* and *pUdt* parameter is passed from the `IOracleCustomType.ToCustomObject` method to the `OracleUdt.IsDBNull` method.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleUdt Class](#)
- [OracleUdt Members](#)

IsDBNull(OracleConnection, object, string)

This method indicates whether or not the attribute being retrieved, specified by `OracleConnection`, UDT, and attribute name, is NULL.

Declaration

```
// C#  
public static bool IsDBNull(OracleConnection con, object Udt, string attrName);
```

Parameters

- *con*
An `OracleConnection` instance.
- *Udt*
An Oracle UDT.
- *attrName*
The case-sensitive name of the attribute.

Return Value

True if the specified attribute is NULL; otherwise, false.

Exceptions

`ArgumentException` - The specified name is not a valid attribute name.

Remarks

This method is invoked from the `IOracleCustomType.ToCustomObject` method. The `con` and `Udt` parameter is passed from the `IOracleCustomType.ToCustomObject` method to the `OracleUdt.IsDBNull` method. The `attrName` parameter is case-sensitive.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleUdt Class](#)
- [OracleUdt Members](#)

IsDBNull(OracleConnection, object, int)

This method indicates whether or not the attribute being retrieved, specified by `OracleConnection`, UDT, and attribute index, is NULL.

Declaration

```
// C#  
public static bool IsDBNull(OracleConnection con, object Udt, int attrIndex);
```

Parameters

- `con`
An `OracleConnection` instance.
- `pUdt`
An Oracle UDT object.
- `attrIndex`
The zero-based index of the attribute.

Return Value

True if the specified attribute is NULL; otherwise, false.

Exceptions

`ArgumentOutOfRangeException` - The specified index is not a valid attribute index

Remarks

This method is invoked from the `IOracleCustomType.ToCustomObject` method. The `con` and `Udt` parameter is passed from the `IOracleCustomType.ToCustomObject` method to the `OracleUdt.IsDBNull` method.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleUdt Class](#)
- [OracleUdt Members](#)

SetValue

SetValue methods set the attributes or elements on the specified Oracle UDT.

Overload List for ODP.NET, Unmanaged Driver:

- [SetValue\(OracleConnection, IntPtr, string, object\)](#)
This method sets the attribute or elements on the specified Oracle UDT, using the specified attribute name and value.
- [SetValue\(OracleConnection, IntPtr, int, object\)](#)
This method sets the attribute or elements on the specified Oracle UDT, using the specified index and value.
- [SetValue\(OracleConnection, IntPtr, string, object, object\)](#)
This method sets either the specified collection attribute of the specified Oracle Object or elements of the specified Oracle Collection, to the specified value using the supplied null status of the collection elements.
- [SetValue\(OracleConnection, IntPtr, int, object, object\)](#)
This method sets either the specified collection attribute of the specified Oracle Object or elements of the specified Oracle Collection, to the specified value using the supplied null status of the collection elements.

Overload List for ODP.NET, Managed Driver and ODP.NET Core

- [SetValue\(OracleConnection, object, string, object\)](#)
This method sets the attribute or elements on the specified Oracle UDT, using the specified attribute name and value.
- [SetValue\(OracleConnection, object, int, object\)](#)
This method sets the attribute or elements on the specified Oracle UDT, using the specified index and value.
- [SetValue\(OracleConnection, object, string, object, object\)](#)
This method sets either the specified collection attribute of the specified Oracle Object or elements of the specified Oracle Collection, to the specified value using the supplied null status of the collection elements.
- [SetValue\(OracleConnection, object, int, object, object\)](#)
This method sets either the specified collection attribute of the specified Oracle Object or elements of the specified Oracle Collection, to the specified value using the supplied null status of the collection elements.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleUdt Class](#)
- [OracleUdt Members](#)

SetValue(OracleConnection, IntPtr, string, object)

This method sets the attribute or elements on the specified Oracle UDT, using the specified attribute name and value.

Declaration

```
// C#
public static void SetValue(OracleConnection con, IntPtr pUdt, string attrName,
    object value);
```

Parameters

- *con*
An OracleConnection instance.
- *pUdt*
An opaque pointer to an Oracle UDT.
- *attrName*
The name of the attribute to be set. Specify null for setting collection elements from a Custom Type that represents an Oracle Collection.
- *value*
The attribute or collection value to be set.

Exceptions

ArgumentException - The specified value is not of the appropriate type.

Remarks

The IOracleCustomType.FromCustomObject method invokes OracleUdt.SetValue method passing it the *con* and *pUdt* parameters. The OracleUdt.SetValue method returns these types of object:

- Oracle Object Type
For a Custom Type that represents an Oracle Object Type, the type accepted for a specified attribute name is the type of the member in the custom class or struct that is mapped to the attribute using the OracleObjectMappingAttribute object.
- Oracle Collection Type
For a Custom Type that represents an Oracle Collection Type, the type accepted is the type of the member in the custom class or struct to which the OracleArrayMappingAttribute object is applied.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleUdt Class](#)
- ["OracleUdt Members"](#)

SetValue(OracleConnection, IntPtr, int, object)

This method sets the attribute or elements on the specified Oracle UDT, using the specified index and value.

Declaration

```
// C#  
public static void SetValue(OracleConnection con, IntPtr pUdt, int attrIndex, object  
value);
```

Parameters

- *con*
An OracleConnection instance.
- *pUdt*
An opaque pointer to an Oracle UDT.
- *attrIndex*
The index of the attribute to be set. Specify 0 for setting collection elements from a Custom Type that represents an Oracle Collection.
- *value*
The attribute or collection value to be set.

Exceptions

ArgumentException - The specified value is not of the appropriate type.

Remarks

The `IOracleCustomType.FromCustomObject` method invokes `OracleUdt.SetValue` method passing it the *con* and *pUdt* parameters. The `OracleUdt.SetValue` method returns these types of object:

- Oracle Object Type
For a Custom Type that represents an Oracle Object Type, the type accepted for a specified attribute index is the type of the member in the custom class or struct that is mapped to the attribute using the `OracleObjectMappingAttribute` object.
- Oracle Collection Type
For a Custom Type that represents an Oracle Collection Type, the type accepted is the type of the member in the custom class or struct to which the `OracleArrayMappingAttribute` object is applied.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleUdt Class](#)
- ["OracleUdt Members"](#)

SetValue(OracleConnection, IntPtr, string, object, object)

This method sets either the specified collection attribute of the specified Oracle Object or elements of the specified Oracle Collection, to the specified value using the supplied null status of the collection elements.

Declaration

```
// C#
public static void SetValue(OracleConnection con, IntPtr pUdt, string attrName,
    object value, object statusArray);
```

Parameters

- *con*
An OracleConnection instance.
- *pUdt*
An opaque pointer to an Oracle UDT.
- *attrName*
The name of the attribute to be set. Specify null for setting collection elements from a Custom Type that represents an Oracle Collection.
- *value*
The attribute or collection value to be set.
- *statusArray*
The null status for the collection elements.

Exceptions

ArgumentException - The specified value is not of the appropriate type.

Remarks

The `IOracleCustomType.FromCustomObject` method invokes `OracleUdt.SetValue` method passing it the *con* and *pUdt* parameters. The `OracleUdt.SetValue` method returns these types of object:

- Oracle Object Type
For a Custom Type that represents an Oracle Object Type, the type accepted for a specified attribute name is the type of the member in the custom class or struct that is mapped to the attribute using the `OracleObjectMappingAttribute` object.
- Oracle Collection Type

For a Custom Type that represents an Oracle Collection Type, the type accepted is the type of the member in the custom class or struct to which the `OracleArrayMappingAttribute` object is applied.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleUdt Class](#)
- ["OracleUdt Members"](#)

SetValue(OracleConnection, IntPtr, int, object, object)

This method sets either the specified collection attribute of the specified Oracle Object or elements of the specified Oracle Collection, to the specified value using the supplied null status of the collection elements.

Declaration

```
// C#  
public static void SetValue(OracleConnection con, IntPtr pUdt, int attrIndex,  
    object value, object statusArray);
```

Parameters

- *con*
An `OracleConnection` instance.
- *pUdt*
An opaque pointer to an Oracle UDT.
- *attrIndex*
The index of the attribute to be set. Specify 0 for setting collection elements from a Custom Type that represents an Oracle Collection.
- *value*
The attribute or collection value to be set.
- *statusArray*
The null status for the collection elements.

Exceptions

`ArgumentException` - The specified value is not of the appropriate type.

Remarks

The `IOracleCustomType.FromCustomObject` method invokes `OracleUdt.SetValue` method passing it the *con* and *pUdt* parameters. The `OracleUdt.SetValue` method returns these types of object:

- Oracle Object Type

For a Custom Type that represents an Oracle Object Type, the type accepted for a specified attribute index is the type of the member in the custom class or struct that is mapped to the attribute using the `OracleObjectMappingAttribute` object.

- Oracle Collection Type

For a Custom Type that represents an Oracle Collection Type, the type accepted is the type of the member in the custom class or struct to which the `OracleArrayMappingAttribute` object is applied.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleUdt Class](#)
- ["OracleUdt Members"](#)

SetValue(OracleConnection, object, string, object)

This method sets the attribute or elements on the specified Oracle UDT, using the specified attribute name and value.

Declaration

```
// C#  
public static void SetValue(OracleConnection con, object Udt, string attrName,  
    object value);
```

Parameters

- *con*
An `OracleConnection` instance.
- *Udt*
An Oracle UDT object.
- *attrName*
The name of the attribute to be set. Specify null for setting collection elements from a Custom Type that represents an Oracle Collection.
- *value*
The attribute or collection value to be set.

Exceptions

`ArgumentException` - The specified value is not of the appropriate type.

Remarks

The `IOracleCustomType.FromCustomObject` method invokes `OracleUdt.SetValue` method passing it the *con* and *Udt* parameters. The `OracleUdt.SetValue` method returns these types of object:

- Oracle Object Type

For a Custom Type that represents an Oracle Object Type, the type accepted for a specified attribute name is the type of the member in the custom class or struct that is mapped to the attribute using the `OracleObjectMappingAttribute` object.

- Oracle Collection Type

For a Custom Type that represents an Oracle Collection Type, the type accepted is the type of the member in the custom class or struct to which the `OracleArrayMappingAttribute` object is applied.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleUdt Class](#)
- ["OracleUdt Members"](#)

SetValue(OracleConnection, object, int, object)

This method sets the attribute or elements on the specified Oracle UDT, using the specified index and value.

Declaration

```
// C#
public static void SetValue(OracleConnection con, object Udt, int attrIndex, object
value);
```

Parameters

- *con*
An `OracleConnection` instance.
- *Udt*
An Oracle UDT object.
- *attrIndex*
The index of the attribute to be set. Specify 0 for setting collection elements from a Custom Type that represents an Oracle Collection.
- *value*
The attribute or collection value to be set.

Exceptions

`ArgumentException` - The specified value is not of the appropriate type.

Remarks

The `IOracleCustomType.FromCustomObject` method invokes `OracleUdt.SetValue` method passing it the *con* and *Udt* parameters. The `OracleUdt.SetValue` method returns these types of object:

- Oracle Object Type

For a Custom Type that represents an Oracle Object Type, the type accepted for a specified attribute index is the type of the member in the custom class or struct that is mapped to the attribute using the `OracleObjectMappingAttribute` object.

- Oracle Collection Type

For a Custom Type that represents an Oracle Collection Type, the type accepted is the type of the member in the custom class or struct to which the `OracleArrayMappingAttribute` object is applied.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleUdt Class](#)
- ["OracleUdt Members"](#)

SetValue(OracleConnection, object, string, object, object)

This method sets either the specified collection attribute of the specified Oracle Object or elements of the specified Oracle Collection, to the specified value using the supplied null status of the collection elements.

Declaration

```
// C#
public static void SetValue(OracleConnection con, object Udt, string attrName,
    object value, object statusArray);
```

Parameters

- *con*
An `OracleConnection` instance.
- *Udt*
An Oracle UDT object.
- *attrName*
The name of the attribute to be set. Specify null for setting collection elements from a Custom Type that represents an Oracle Collection.
- *value*
The attribute or collection value to be set.
- *statusArray*
The null status for the collection elements.

Exceptions

`ArgumentException` - The specified value is not of the appropriate type.

Remarks

The `IOracleCustomType.FromCustomObject` method invokes `OracleUdt.SetValue` method passing it the *con* and *Udt* parameters. The `OracleUdt.SetValue` method returns these types of object:

- Oracle Object Type

For a Custom Type that represents an Oracle Object Type, the type accepted for a specified attribute name is the type of the member in the custom class or struct that is mapped to the attribute using the `OracleObjectMappingAttribute` object.

- Oracle Collection Type

For a Custom Type that represents an Oracle Collection Type, the type accepted is the type of the member in the custom class or struct to which the `OracleArrayMappingAttribute` object is applied.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleUdt Class](#)
- ["OracleUdt Members"](#)

SetValue(OracleConnection, object, int, object, object)

This method sets either the specified collection attribute of the specified Oracle Object or elements of the specified Oracle Collection, to the specified value using the supplied null status of the collection elements.

Declaration

```
// C#
public static void SetValue(OracleConnection con, object Udt, int attrIndex,
    object value, object statusArray);
```

Parameters

- *con*
An `OracleConnection` instance.
- *Udt*
An Oracle UDT object.
- *attrIndex*
The index of the attribute to be set. Specify 0 for setting collection elements from a Custom Type that represents an Oracle Collection.
- *value*
The attribute or collection value to be set.
- *statusArray*
The null status for the collection elements.

Exceptions

ArgumentException - The specified value is not of the appropriate type.

Remarks

The `IOracleCustomType.FromCustomObject` method invokes `OracleUdt.SetValue` method passing it the *con* and *Udt* parameters. The `OracleUdt.SetValue` method returns these types of object:

- Oracle Object Type

For a Custom Type that represents an Oracle Object Type, the type accepted for a specified attribute index is the type of the member in the custom class or struct that is mapped to the attribute using the `OracleObjectMappingAttribute` object.

- Oracle Collection Type

For a Custom Type that represents an Oracle Collection Type, the type accepted is the type of the member in the custom class or struct to which the `OracleArrayMappingAttribute` object is applied.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleUdt Class](#)
- "[OracleUdt Members](#)"

OracleRef Class

An `OracleRef` instance represents an Oracle REF, which references a persistent, standalone, referenceable object that resides in the database. The `OracleRef` object provides methods to insert, update, and delete the Oracle REF.

Class Inheritance

`System.Object`

`System.MarshalByRefObject`

`Oracle.DataAccess.Types.OracleRef`

Declaration

```
// C#
public sealed class OracleRef : MarshalByRefObject, ICloneable, IDisposable,
    INullable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

If two or more OracleRef objects that refer to the same Oracle object in the database are retrieved through the same OracleConnection, then their operations on the referenced object must be synchronized.

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Members](#)
- [OracleRef Constructors](#)
- [OracleRef Static Methods](#)
- [OracleRef Instance Properties](#)
- [Oracle Ref Instance Methods](#)

OracleRef Members

OracleRef members are listed in the following tables.

OracleRef Constructors

OracleRef constructors are listed in [Table 16-30](#).

Table 16-30 OracleRef Constructors

Constructor	Description
OracleRef Constructors	Instantiates a new instance of OracleRef class (Overloaded) <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>

OracleRef Static Fields

OracleRef static methods are listed in [Table 16-31](#)

Table 16-31 OracleRef Static Fields

Static Field	Description
Null	Represents a null value that can be assigned to an OracleRef instance

OracleRef Static Methods

OracleRef static methods are listed in [Table 16-32](#).

Table 16-32 OracleRef Static Methods

Method	Description
Equals	Inherited from System.Object (Overloaded)

OracleRef Instance Properties

OracleRef instance properties are listed in [Table 16-33](#).

Table 16-33 OracleRef Instance Properties

Property	Description
Connection	References the connection used by the OracleRef
HasChanges	References the connection used by the OracleRef <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
IsLocked	Indicates whether or not the REF is locked
IsNull	Indicates whether or not the Oracle REF is NULL
ObjectTableName	Returns the fully qualified object table name that is associated with the REF <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
Value	Returns a .NET representation of this Oracle REF

OracleRef Instance Methods

OracleRef instance methods are listed in [Table 16-34](#).

Table 16-34 OracleRef Instance Methods

Method	Description
Clone	Clones the REF
Delete	Deletes the referenced object from the database See also Table 3-37
Dispose	Releases resources allocated for the OracleRef instance
Equals	Inherited from System.Object

Table 16-34 (Cont.) OracleRef Instance Methods

Method	Description
Flush	Flushes changes made on the REF object to the database <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
GetCustomObject	Returns the object that the specified REF references as a custom type (Overloaded) See also Table 3-37
GetCustomObjectForUpdate	Returns the object that the specified REF references as a custom type (Overloaded) <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
GetHashCode	Inherited from System.Object
GetType	Inherited from System.Object
IsEqual	Compares two OracleRef objects
Lock	Locks the REF in the database <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
ToString	Inherited from System.Object
Update	Updates the object referenced by the specified REF in the database using the specified custom object See also Table 3-37

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Class](#)

OracleRef Constructors

OracleRef constructors instantiate new instances of OracleRef class.

Overload List:

- [OracleRef\(OracleConnection, string\)](#)
This constructor creates an instance of the OracleRef class with a connection and a HEX string that represents an REF instance in the database.
Not Available in ODP.NET, Managed Driver and ODP.NET Core
- [OracleRef\(OracleConnection, string, string\)](#)
This constructor creates an instance of the OracleRef class using the specified OracleConnection object, user-defined type name, and an object table name
Not Available in ODP.NET, Managed Driver and ODP.NET Core

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Class](#)
- [OracleRef Members](#)

OracleRef(OracleConnection, string)

This constructor creates an instance of the `OracleRef` class with a connection and a HEX string that represents an REF instance in the database.

Declaration

```
// C#  
public OracleRef(OracleConnection con, string hexStr);
```

Parameters

- *con*
An `OracleConnection` instance.
- *hexStr*
A HEX string that represents an REF instance in the database.

Exceptions

`ArgumentException` - The HEX string does not represent a valid REF in the database.

`ArgumentNullException` - The connection or HEX string is null.

`InvalidOperationException` - The `OracleConnection` object is not open.

Remarks

When an `OracleRef` instance is created, it is referenced to a specific table in the database.

The connection must be opened explicitly by the application. `OracleRef` does not open the connection implicitly.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Class](#)
- [OracleRef Members](#)

OracleRef(OracleConnection, string, string)

This constructor creates an instance of the `OracleRef` class using the specified `OracleConnection` object, user-defined type name, and an object table name.

Declaration

```
// C#  
public OracleRef(OracleConnection con, string udtTypeName, string objTabName);
```

Parameters

- *con*
An `OracleConnection` instance.
- *udtTypeName*
A user-defined type name.
- *objTabName*
An object table name.

Exceptions

`ArgumentException` - The object type name or the object table name is not valid.

`ArgumentNullException` - The object type name or the table name is null.

`InvalidOperationException` - The `OracleConnection` object is not open.

Remarks

When an `OracleRef` instance is created, this `OracleRef` instance is associated with the specific table in the database. In other words, it represents a persistent REF.

This constructor creates a reference to the object table. However, it does not cause any entries to be made in database tables until the object is flushed to the database, that is, until the `OracleRef.Flush` or the `OracleConnection.FlushCache` method is called on the `OracleRef` Connection. Therefore, any operation that attempts to operate on the database copy of the object before flushing the object, such as, lock the object or fetch the latest copy of the object from the database, results in an `OracleException`.

The connection must be opened explicitly by the application. `OracleRef` does not open the connection implicitly.

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleRef Class](#)
- [OracleRef Members](#)
- "[FlushCache](#)"

OracleRef Static Fields

`OracleRef` static fields are listed in [Table 16-35](#).

Table 16-35 OracleRef Static Fields

Static Field	Description
Null	Represents a null value that can be assigned to an OracleRef instance

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Class](#)
- [OracleRef Members](#)

Null

This static field represents a null value that can be assigned to an OracleRef instance.

Declaration

```
// C#  
public static readonly OracleRef Null;
```

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Class](#)
- [OracleRef Members](#)

OracleRef Static Methods

OracleRef static methods are listed in [Table 16-36](#).

Table 16-36 OracleRef Static Methods

Method	Description
Equals	Inherited from System.Object (Overloaded)

 See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Class](#)
- [OracleRef Members](#)

OracleRef Instance Properties

OracleRef instance properties are listed in [Table 16-37](#).

Table 16-37 OracleRef Instance Properties

Property	Description
Connection	References the connection used by the OracleRef
HasChanges	References the connection used by the OracleRef <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
IsLocked	Indicates whether or not the REF is locked
IsNull	Indicates whether or not the Oracle REF is NULL
ObjectTableName	Returns the fully qualified object table name that is associated with the REF <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
Value	Returns a .NET representation of this Oracle REF

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Class](#)
- [OracleRef Members](#)

Connection

This instance property references the connection used by the OracleRef.

Declaration

```
// C#  
public OracleConnection Connection{get;}
```

Property Value

An OracleConnection object associated with the REF.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

Once the Dispose method is invoked, this property is set to null.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Class](#)
- [OracleRef Members](#)

HasChanges

This instance property indicates whether or not the object referenced by the Oracle REF in the object cache has any changes that can be flushed to the database.

Declaration

```
// C#  
public bool HasChanges {get;}
```

Property Value

Returns true if the object referenced by the Oracle REF in the object cache has any changes that can be flushed to the database; otherwise, returns false.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

This property returns true if a copy of the referenced object in the object cache is updated or deleted.

If there is no copy of the referenced object in the object cache, the latest copy of the referenced object in the database is cached in the object cache and false is returned.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Class](#)
- [OracleRef Members](#)

IsLocked

This instance property indicates whether or not the REF is locked.

Declaration

```
// C#  
public bool IsLocked {get;}
```

Property Value

Returns true if the REF is locked; otherwise returns false.

Exceptions

ObjectDisposedException - The object is already disposed.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Class](#)
- [OracleRef Members](#)

IsNull

This instance property indicates whether or not the Oracle REF is NULL.

Declaration

```
// C#  
public bool IsNull {get;}
```

Property Value

Returns true if the REF is NULL; otherwise, returns false.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

If the Oracle REF is NULL, this property returns true. Otherwise, it returns false.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Class](#)
- [OracleRef Members](#)

ObjectName

This instance property returns the fully-qualified object table name that is associated with the REF.

Declaration

```
// C#  
public string ObjectTableName{get;}
```


Property Value

A fully-qualified object table name that is associated with the REF.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

The object table name is in the form *schema_Name.Table_Name*.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Class](#)
- [OracleRef Members](#)

Value

This instance property returns a .NET representation of this Oracle REF.

Declaration

```
// C#  
public string Value{get;}
```

Property Value

A .NET representation of the Oracle REF.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

This property returns a HEX string that represents the REF.

The returned string can be used to create a new OracleRef instance by using the OracleRef(OracleConnection, string) constructor.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Class](#)
- [OracleRef Members](#)
- ["OracleRef\(OracleConnection, string\)"](#)

Oracle Ref Instance Methods

OracleRef instance methods are listed in [Table 16-38](#).

Table 16-38 OracleRef Instance Methods

Method	Description
Clone	Clones the REF
Delete	Deletes the referenced object from the database See also Table 3-37
Dispose	Releases resources allocated for the OracleRef instance
Equals	Inherited from System.Object
Flush	Flushes changes made on the REF object to the database <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
GetCustomObject	Returns the object that the specified REF references as a custom type (Overloaded) See also Table 3-37
GetCustomObjectForUpdate	Returns the object that the specified REF references as a custom type (Overloaded) <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
GetHashCode	Inherited from System.Object
GetType	Inherited from System.Object
IsEqual	Compares two OracleREF objects
Lock	Locks the REF in the database <i>Not Available in ODP.NET, Managed Driver and ODP.NET Core</i>
ToString	Inherited from System.Object
Update	Updates the object referenced by the specified REF in the database using the specified custom object See also Table 3-37

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Class](#)
- [OracleRef Members](#)

Clone

This instance method clones the REF.

Declaration

```
// C#
public OracleRef Clone();
```

Return Value

A clone of the current instance.

Implements

ICloneable

Exceptions

InvalidOperationException - The associated connection is not open.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Class](#)
- [OracleRef Members](#)

Delete

This method deletes the referenced object from the database.

Declaration

```
// C# - managed and core ODP.NET  
public void Delete();
```

```
// C# - unmanaged ODP.NET  
public void Delete(bool bFlush);
```

Parameters

- *bFlush*
A bool that specifies whether or not the REF is flushed immediately.

Remarks

This method marks the specified REF for deletion.

In managed ODP.NET and ODP.NET Core, Delete() has the same behavior as unmanaged ODP.NET Delete(true).

Depending on whether the value of *bFlush* is set to true or false, the following occurs:

- True
The object referenced by the specified REF is deleted immediately from the database.
Before flushing objects, it is required that the application has explicitly started a transaction by executing the BeginTransaction method on the OracleConnection object. This is because if the object being flushed has not already been locked by the application, an exclusive lock is obtained implicitly for the object. The lock is only released when the transaction commits or rollbacks.
- False

The object referenced by the REF is not deleted immediately from the database, but only when a subsequent Flush method is invoked for the specified REF or the FlushCache method is invoked on the OracleRef or the FlushCache method is invoked on the OracleRef connection.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Class](#)
- [OracleRef Members](#)
- OracleConnection ["FlushCache"](#)

Dispose

This instance method releases resources allocated for the OracleRef instance.

Declaration

```
// C#  
public void Dispose();
```

Implements

IDisposable

Remarks

The object cannot be reused after it is disposed. Although some properties can still be accessed, their values may not be up-to-date.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Class](#)
- [OracleRef Members](#)

Flush

This instance method flushes changes made on the REF object to the database, such as updates or deletes.

Declaration

```
// C#  
public void Flush();
```

Exceptions

InvalidOperationException - The associated connection is not open.

Remarks

Before flushing objects, it is required that the application has explicitly started a transaction by executing the `BeginTransaction` method on the `OracleConnection` object. This is because if the object being flushed has not already been locked by the application, an exclusive lock is obtained implicitly for the object. The lock is only released when the transaction commits or rollbacks.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Class](#)
- [OracleRef Members](#)

GetCustomObject

`GetCustomObject` methods return the object that the specified REF references as a custom type.

Overload List

- [GetCustomObject\(OracleUdtFetchOption\)](#)
This method returns the object that the specified REF references as a custom type using the specified fetch option.
- [GetCustomObject\(OracleUdtFetchOption, int\)](#)
This method returns the object that the specified REF references as a custom type using the specified fetch option and depth level.

Not supported in ODP.NET, Managed Driver and ODP.NET Core.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Class](#)
- [OracleRef Members](#)

GetCustomObject(OracleUdtFetchOption)

This method returns the object that the specified REF references, as a custom type, using the specified fetch option.

Declaration

```
// C# - managed and core ODP.NET
public object GetCustomObject();
```

```
// C# - unmanaged ODP.NET  
public object GetCustomObject(OracleUdtFetchOption fetchOption);
```

Parameters

- *fetchOption*
An `OracleUdtFetchOption` value.

Return Value

A custom object that represents the object that the specified REF references.

Exceptions

`InvalidOperationException` - The specified connection is not open, or a valid custom type has not been registered for the type of the referenced object.

Remarks

This method returns a custom type determined by the UDT mappings on the specified connection.

The connection must be opened explicitly by the application. This method does not open the connection implicitly.

Managed ODP.NET and ODP.NET Core always retrieves the object from the database as it does not have an object cache.

The application can use the `OracleUdtFetchOption` method to control the copy of the Object that is returned according to the specified option:

- `OracleUdtFetchOption.Cache` option
If this option is specified, and there is a copy of the referenced object in the object cache, it is returned immediately. If no cached copy exists, the latest copy of the referenced object in the database is cached in the object cache and returned.
- `OracleUdtFetchOption.Server` option
If this option is specified, the latest copy of the referenced object from the database is cached in the object cache and returned. If a copy of the referenced object already exists in the cache, the latest copy overwrites the existing one.
- `OracleUdtFetchOption.TransactionCache` option
If this option is specified, and a copy of the referenced object is cached in the current transaction, the copy is returned. Otherwise, the latest copy of the referenced object from the database is cached in the object cache and returned. If a copy of the referenced object already exists in the cache, the latest copy overwrites the existing one.

Note that if a cached copy of the referenced object was modified before the current transaction began, that is, if the `OracleRef.HasChanges` property returns `true`, then the `Recent` option returns the cached copy of the referenced object. Outside of a transaction, the `Recent` option behaves like the `Any` option.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Class](#)
- [OracleRef Members](#)
- ["OracleUdtFetchOption Enumeration"](#)

GetCustomObject(OracleUdtFetchOption, int)

This method returns the object that the specified REF references, as a custom type, using the specified fetch option and depth level.

Declaration

```
// C#  
public object GetCustomObject(OracleUdtFetchOption fetchOption, int depthLevel);
```

Parameters

- *fetchOption*
An `OracleUdtFetchOption` value.
- *depthLevel*
The number of levels to be fetched for nested REF attributes.

Return Value

A custom object that represents the object that the specified REF references.

Exceptions

`InvalidOperationException` - The specified connection is not open, or a valid custom type has not been registered for the type of the referenced object.

Remarks

This method returns a custom type determined by the UDT mappings on the specified connection.

If the object that the REF references contains nested REF attributes, the *depthLevel* can be specified to optimize the subsequent object retrieval. The value of *depthLevel* determines the number of levels that are optimized.

For example, if the *depthLevel* is specified as two, the optimization is applied to all top-level nested REF attributes in the object being fetched and also to all nested REF attributes within the objects referenced by the top-level nested REF attributes.

The connection must be opened explicitly by the application. This method does not open the connection implicitly.

The application can use the `OracleUdtFetchOption` method to control the copy of the Object that is returned according to the specified option:

- `OracleUdtFetchOption.Cache` option

If this option is specified, and there is a copy of the referenced object in the object cache, it is returned immediately. If no cached copy exists, the latest copy of the referenced object in the database is cached in the object cache and returned.

- `OracleUdtFetchOption.Server` option

If this option is specified, the latest copy of the referenced object from the database is cached in the object cache and returned. If a copy of the referenced object already exists in the cache, the latest copy overwrites the existing one.

- `OracleUdtFetchOption.TransactionCache` option

If this option is specified, and a copy of the referenced object is cached in the current transaction, the copy is returned. Otherwise, the latest copy of the referenced object from the database is cached in the object cache and returned. If a copy of the referenced object already exists in the cache, the latest copy overwrites the existing one.

Note that if a cached copy of the referenced object was modified before the current transaction began, that is, if the `OracleRef.HasChanges` property returns `true`, then the `Recent` option returns the cached copy of the referenced object. Outside of a transaction, the `Recent` option behaves like the `Any` option.

① See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleRef Class](#)
- [OracleRef Members](#)
- "[OracleUdtFetchOption Enumeration](#)"

GetCustomObjectForUpdate

`GetCustomObjectForUpdate` methods return the object that the specified REF references as a custom type.

- [GetCustomObjectForUpdate\(bool\)](#)

This method locks the specified REF in the database and returns the object that the specified REF references as a custom type using the specified wait option.

Not supported in ODP.NET, Managed Driver and ODP.NET Core.

- [GetCustomObjectForUpdate\(bool, int\)](#)

This method locks the specified REF in the database and returns the object that the specified REF references as a custom type using the specified wait option and depth level.

Not supported in ODP.NET, Managed Driver and ODP.NET Core.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Class](#)
- [OracleRef Members](#)

GetCustomObjectForUpdate(bool)

This method locks the specified REF in the database and returns the object that the specified REF references, as a custom type, using the specified wait option.

Declaration

```
// C#  
public object GetCustomObjectForUpdate(bool bWait);
```

Parameters

- *bWait*

Specifies if the REF is to be locked with the no-wait option. If wait is set to true, this method invocation does not return until the REF is locked.

Return Value

A custom object that represents the object that the specified REF references.

Exceptions

InvalidOperationException - The specified connection is not open, or a valid custom type has not been registered for type of the referenced object.

OracleException - *bWait* is set to false, and the lock cannot be acquired.

Remarks

This method returns the latest copy of the referenced object, as a custom type, determined by the custom types registered on the *OracleRef* connection.

To be able to release the lock on the REF appropriately after flushing the REF using the *Flush* method on the *OracleRef* or *FlushCache* method on the *OracleConnection*, the application must commit or rollback the transaction. Therefore, it is required that, before invoking this method, a transaction is explicitly started by executing the *BeginTransaction* method on the *OracleConnection* object.

This method makes a network round-trip to lock the REF in the database. After this call, programmers can modify the associated row object exclusively. Then a call to the *Flush* method on the *OracleRef* or *FlushCache* method on the *OracleConnection* flushes the changes to the database.

If true is passed, this method blocks until the lock can be acquired. If false is passed, this method immediately returns. If the lock cannot be acquired, an *OracleException* is thrown.

The connection must be opened explicitly by the application. This method does not open the connection implicitly.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Class](#)
- [OracleRef Members](#)

GetCustomObjectForUpdate(bool, int)

This method locks the specified REF in the database and returns the object that the specified REF references, as a custom type, using the specified wait option and depth level

Declaration

```
public object GetCustomObjectForUpdate(bool bWait, int depthLevel);
```

Parameters

- *bWait*
A boolean value that specifies if the REF is to be locked with the no-wait option. If wait is set to true, this method invocation does not return until the REF is locked.
- *depthLevel*
The number of levels to be fetched for nested REF attributes.

Return Value

A custom object that represents the object that the specified REF references.

Exceptions

`InvalidOperationException` - The specified connection is not open, or a valid custom type has not been registered for type of the referenced object.

`OracleException` - *bWait* is set to false, and the lock cannot be acquired.

Remarks

This method returns the latest copy of the referenced object, as a custom type, determined by the custom types registered on the `OracleRef` connection.

To be able to release the lock on the REF appropriately after flushing the REF using the `Flush` method on the `OracleRef` or `FlushCache` method on the `OracleConnection`, the application must commit or rollback the transaction. Therefore, it is required that, before invoking this method, a transaction is explicitly started by executing the `BeginTransaction` method on the `OracleConnection` object.

This method makes a network round-trip to lock the REF in the database. After this call, programmers can modify the associated row object exclusively. Then a call to the `Flush` method on the `OracleRef` or `FlushCache` method on the `OracleConnection` flushes the changes to the database.

If `true` is passed, this method blocks until the lock can be acquired. If `false` is passed, this method immediately returns. If the lock cannot be acquired, an `OracleException` is thrown.

If the object that the REF references contains nested REF attributes, the *depthLevel* can be specified to optimize the subsequent object retrieval. The value of *depthLevel* determines the number of levels that are optimized.

For example, if the *depthLevel* is specified as 2, the optimization is applied to all top-level nested REF attributes in the object being fetched and also to all nested REF attributes within the objects referenced by the top-level nested REF attributes.

The connection must be opened explicitly by the application. This method does not open the connection implicitly.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Class](#)
- [OracleRef Members](#)

IsEqual

This instance method compares two OracleRef objects.

Declaration

```
// C#  
public bool IsEqual(OracleRef oraRef);
```

Parameters

- *oraRef*
The provided OracleRef object.

Return Value

bool

Remarks

This instance method returns true if the OracleRef instance and the OracleRef parameter both reference the same object. Otherwise, it returns false.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Class](#)
- [OracleRef Members](#)

Lock

This instance method locks the REF in the database.

Declaration

```
// C#  
public bool Lock(bool bWait);
```

Parameters

- *bWait*

Specifies if the lock is set to the no-wait option. If *bWait* is set to `true`, the method invocation does not return until the REF is locked.

Return Value

A boolean value that indicates whether or not the lock has been acquired.

Exceptions

`InvalidOperationException` - The associated connection is not open.

`ObjectDisposedException` - The object is already disposed.

Remarks

In order for the application to release the lock on the REF appropriately after the `Flush` invocation on the `OracleRef` or `FlushCache` methods, the application must commit or rollback the transaction. Therefore, it is required that, before invoking a lock on an `OracleRef` object, a transaction is explicitly started by executing the `BeginTransaction` method on the `OracleConnection` object.

This instance method makes a network round-trip to lock the REF in the database. After this call, programmers can modify the attribute values of the associated row object exclusively. Then a call to the `Flush` instance method on the `OracleRef` or `FlushCache` method on the `OracleConnection` flushes the changes to the database.

If `true` is passed, this method blocks, that is, does not return, until the lock is acquired. Consequently, the return value is always `true`.

If `false` is passed, this method immediately returns. The return value indicates `true` if the lock is acquired, and `false` if it is not.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Class](#)
- [OracleRef Members](#)

Update

This method updates the object referenced by the specified REF in the database using the specified custom object.

Declaration

```
// C# - managed and core ODP.NET
public void Update(object customObject);

// C# - unmanaged ODP.NET
public void Update(object customObject, bool bFlush);
```

Parameters

- *customObject*
The custom object used to update the referenced object.
- *bFlush*
A boolean that specifies if the changes must be flushed immediately. If *bFlush* is set to true, this method invocation flushes the changes immediately.

Exceptions

`InvalidOperationException` - The specified connection is not open or the custom object does not map to the type of referenced object.

Remarks

This method marks the specified REF for update. In managed ODP.NET and ODP.NET Core, `Update(customObject)` has the same behavior as unmanaged ODP.NET `Update(customObject, true)`. Depending on whether the value of *bFlush* is set to true or false, the following occurs:

- True
The object referenced by the specified REF is updated immediately in the database.
Before flushing objects, it is required that the application has explicitly started a transaction by executing the `BeginTransaction` method on the `OracleConnection` object. This is because if the object being flushed has not already been locked by the application, an exclusive lock is obtained implicitly for the object. The lock is only released when the transaction commits or rollbacks.
- False
The object referenced by the REF is not updated immediately in the database, but only when a subsequent `Flush` method is invoked for the specified REF or the `FlushCache` method is invoked for the specified connection.

The connection must be opened explicitly by the application. This method does not open the connection implicitly.

See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Class](#)
- [OracleRef Members](#)

OracleUdtFetchOption Enumeration

OracleUdtFetchOption enumeration values specify how to retrieve a copy of the referenceable object.

[Table 16-39](#) lists all the OracleUdtFetchOption enumeration values with a description of each enumerated value.

Table 16-39 OracleUdtFetchOption Enumeration Values

Member Name	Description
Cache	If there is a copy of the referenced object in the object cache, it is returned immediately. If no cached copy exists, the latest copy of the referenced object in the database is cached in the object cache and returned.
Server	The latest copy of the referenced object from the database is cached in the object cache and returned. If a copy of the referenced object already exists in the cache, the latest copy overwrites the existing one.
TransactionCache	If a copy of the referenced object is cached in the current transaction, the copy is returned. Otherwise, the latest copy of the referenced object from the database is cached in the object cache and returned. If a copy of the referenced object already exists in the cache, the latest copy overwrites the existing one. Note that if a cached copy of the referenced object was modified before the current transaction began, that is, if the <code>OracleRef.HasChanges</code> property returns <code>true</code>, then the <code>Recent</code> option returns the cached copy of the referenced object. Outside of a transaction, the <code>Recent</code> option behaves like the <code>Any</code> option.

① See Also

- ["Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"](#)
- [OracleRef Class](#)
- [OracleRef Members](#)

OracleUdtStatus Enumeration

OracleUdtStatus enumeration values specify the status of an object attribute or collection element. An object attribute or a collection element can be a valid value or a null value.

[Table 16-40](#) lists all the OracleUdtStatus enumeration values with a description of each enumerated value:

Table 16-40 OracleUdtStatus Enumeration Values

Member Name	Description
Null	Indicates that an object attribute or collection element is NULL.

Table 16-40 (Cont.) OracleUdtStatus Enumeration Values

Member Name	Description
NotNull	Indicates that a non-NULL value exists for the object attribute or collection element.

See Also

- "[Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces](#)"
- [OracleRef Class](#)
- [OracleRef Members](#)

Oracle Data Provider for .NET Bulk Copy Classes

This chapter describes Oracle Data Provider for .NET support for Bulk Copy operations.

Note

Oracle Data Provider for .NET bulk copy operations do not support loading of UDT type columns.

This chapter includes the following topics:

- [OracleBulkCopy Class](#)
- [OracleBulkCopyColumnMapping Class](#)
- [OracleBulkCopyColumnMappingCollection Class](#)
- [OracleBulkCopyOptions Enumeration](#)
- [OracleRowsCopiedEventHandler Delegate](#)
- [OracleRowsCopiedEventArgs Class](#)

OracleBulkCopy Class

An `OracleBulkCopy` object efficiently bulk loads or copies data into an Oracle table from another data source.

Class Inheritance

`System.Object`

`System.OracleBulkCopy`

Declaration

```
// C#
public sealed class OracleBulkCopy : IDisposable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

The OracleBulkCopy class can be used to write data to Oracle database tables only. However, the data source is not limited to Oracle databases; any data source can be used, as long as the data can be loaded to a DataTable instance or read with an IDataReader instance.

Bulk copy of string data to destination number column is currently not supported.

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleBulkCopy Members](#)
- [OracleBulkCopy Constructors](#)
- [OracleBulkCopy Properties](#)
- [OracleBulkCopy Public Methods](#)
- [OracleBulkCopy Events](#)

OracleBulkCopy Members

OracleBulkCopy members are listed in the following tables.

OracleBulkCopy Constructors

OracleBulkCopy constructors are listed in [Table 17-1](#).

Table 17-1 OracleBulkCopy Constructors

Constructor	Description
OracleBulkCopy Constructors	OracleBulkCopy constructors create new instances of the OracleBulkCopy class

OracleBulkCopy Properties

OracleBulkCopy properties are listed in [Table 17-2](#).

Table 17-2 OracleBulkCopy Properties

Property	Description
BatchSize	Specifies the number of rows to be sent as a batch to the database
BulkCopyOptions	Specifies the <code>OracleBulkCopyOptions</code> enumeration value that determines the behavior of the bulk copy operation
BulkCopyTimeout	Specifies the number of seconds allowed for the bulk copy operation to complete before it is aborted
ColumnMappings	Specifies the column mappings between the data source and destination table
Connection	Specifies the <code>OracleConnection</code> object that the Oracle database uses to perform the bulk copy operation
DestinationPartitionName	Specifies the database partition that the data is loaded into
DestinationSchemaName	Specifies the database schema that the data is loaded into
DestinationTableName	Specifies the database table that the data is loaded in
NotifyAfter	Defines the number of rows to be processed before a notification event is generated

OracleBulkCopy Public Methods

OracleBulkCopy public methods are listed in [Table 17-3](#).

Table 17-3 OracleBulkCopy Public Methods

Method	Description
Close	Closes the <code>OracleBulkCopy</code> instance
Dispose	Releases any resources or memory allocated by the object
WriteToServer	Copies rows to a destination table

OracleBulkCopy Events

OracleBulkCopy events are listed in [Table 17-4](#).

Table 17-4 OracleBulkCopy Events

Event	Description
OracleRowsCopied	Triggered every time the number of rows specified by the <code>OracleBulkCopy.NotifyAfter</code> property has been processed

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopy Class](#)

OracleBulkCopy Constructors

OracleBulkCopy constructors create new instances of the OracleBulkCopy class.

Overload List:

- [OracleBulkCopy\(OracleConnection\)](#)
This constructor instantiates a new instance of OracleBulkCopy class using the specified connection and default value for OracleBulkCopyOptions.
- [OracleBulkCopy\(string\)](#)
This constructor instantiates a new instance of OracleBulkCopy based on the supplied *connectionString* and default value for OracleBulkCopyOptions.
- [OracleBulkCopy\(OracleConnection, OracleBulkCopyOptions\)](#)
This constructor instantiates a new instance of OracleBulkCopy using the specified connection object and OracleBulkCopyOptions value.
- [OracleBulkCopy\(string, OracleBulkCopyOptions\)](#)
This constructor instantiates a new instance of OracleConnection based on the supplied *connectionString* and OracleBulkCopyOptions value.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopy Class](#)
- [OracleBulkCopy Members](#)

OracleBulkCopy(OracleConnection)

This constructor instantiates a new instance of OracleBulkCopy class using the specified connection and default OracleBulkCopyOptions enumeration values.

Declaration

```
// C#  
public OracleBulkCopy(OracleConnection connection);
```

Parameters

- *connection*
The open instance of OracleConnection that performs the bulk copy operation.

Exceptions

ArgumentNullException - The connection parameter is null.

InvalidOperationException - The connection is not in the open state.

Remarks

The connection object passed to this constructor must be open. It remains open after the `OracleBulkCopy` instance is closed.

This constructor uses the default enumeration value `OracleBulkCopyOptions.Default`.

The `Connection` property is set to the supplied connection.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopy Class](#)
- [OracleBulkCopy Members](#)
- ["OracleBulkCopyOptions Enumeration"](#)

OracleBulkCopy(string)

This constructor instantiates a new instance of the `OracleBulkCopy` class by first creating an `OracleConnection` object based on the supplied *connectionString*, then initializing the new `OracleBulkCopy` object with the `OracleConnection` object and `OracleBulkCopyOptions` default value.

Declaration

```
// C#  
public OracleBulkCopy(string connectionString);
```

Parameters

- *connectionString*

The connection information used to connect to the Oracle database and perform the bulk copy operation.

Exception

`ArgumentNullException` - The *connectionString* parameter is null.

`ArgumentException` - The *connectionString* parameter is empty.

Remarks

The `WriteToServer` method opens the connection, if it is not already opened. The connection is automatically closed when the `OracleBulkCopy` instance is closed.

This constructor uses the default enumeration value `OracleBulkCopyOptions.Default`.

The `Connection` property is set to the `OracleConnection` object initialized using the supplied *connectionString*.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopy Class](#)
- [OracleBulkCopy Members](#)
- ["OracleBulkCopyOptions Enumeration"](#)

OracleBulkCopy(OracleConnection, OracleBulkCopyOptions)

This constructor instantiates a new instance of `OracleBulkCopy` using the specified connection object and `OracleBulkCopyOptions` value.

Declaration

```
// C#  
public OracleBulkCopy(OracleConnection connection, OracleBulkCopyOptions  
    copyOptions);
```

Parameters

- *connection*
The open instance of an `OracleConnection` object that performs the bulk copy operation.
- *copyOptions*
The combination of `OracleBulkCopyOptions` enumeration values that determine the behavior of the `OracleBulkCopy` object.

Exceptions

`ArgumentNullException` - The *connection* parameter is null.

`InvalidOperationException` - The connection is not in the open state.

Remarks

The connection passed to this constructor must be open. It remains open after the `OracleBulkCopy` instance is closed.

The `Connection` property is set to the supplied connection.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopy Class](#)
- [OracleBulkCopy Members](#)
- ["OracleBulkCopyOptions Enumeration"](#)

OracleBulkCopy(string, OracleBulkCopyOptions)

This constructor instantiates a new instance of the `OracleBulkCopy` class by first creating an `OracleConnection` object based on the supplied *connectionString*, then initializing the new `OracleBulkCopy` object with the `OracleConnection` object and the supplied `OracleBulkCopyOptions` enumeration values.

Declaration

```
// C#  
public OracleBulkCopy(string connectionString, OracleBulkCopyOptions copyOptions);
```

Parameters

- *connectionString*
The connection information used to connect to the Oracle database to perform the bulk copy operation.
- *copyOptions*
The combination of `OracleBulkCopyOptions` enumeration values that determine the behavior of the bulk copy operation.

Exceptions

`ArgumentNullException` - The *connectionString* is null.

`ArgumentException` - The *connectionString* parameter is empty.

Remarks

The constructor uses the new instance of the `OracleConnection` class to initialize a new instance of the `OracleBulkCopy` class. The `OracleBulkCopy` instance behaves according to options supplied in the *copyOptions* parameter.

The connection is automatically closed when the `OracleBulkCopy` instance is closed.

The `Connection` property is set to an `OracleConnection` object initialized using the supplied *connectionString*.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopy Class](#)
- [OracleBulkCopy Members](#)
- ["OracleBulkCopyOptions Enumeration"](#)

OracleBulkCopy Properties

`OracleBulkCopy` properties are listed in [Table 17-5](#).

Table 17-5 OracleBulkCopy Properties

Property	Description
BatchSize	Specifies the number of rows to be sent as a batch to the database
BulkCopyOptions	Specifies the <code>OracleBulkCopyOptions</code> enumeration value that determines the behavior of the bulk copy operation
BulkCopyTimeout	Specifies the number of seconds allowed for the bulk copy operation to complete before it is aborted
ColumnMappings	Specifies the column mappings between the data source and destination table
Connection	Specifies the <code>OracleConnection</code> object that the Oracle database uses to perform the bulk copy operation
DestinationPartitionName	Specifies the database partition that the data is loaded into
DestinationSchemaName	Specifies the database schema that the data is loaded into
DestinationTableName	Specifies the database table that the data is loaded in
NotifyAfter	Defines the number of rows to be processed before a notification event is generated

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopy Class](#)
- [OracleBulkCopy Members](#)

BatchSize

This property specifies the number of rows to be sent as a batch to the database.

Declaration

```
// C#  
public int BatchSize {get; set;}
```

Property Value

An integer value for the number of rows to be sent to the database as a batch.

Exceptions

`ArgumentOutOfRangeException` - The batch size is less than zero.

Remarks

The default value is zero, indicating that the rows are not sent to the database in batches. The entire set of rows are sent in one single batch.

A large batch size reduces database round trips, but it can also consume large amounts of client side memory. Excessive memory consumption slows down overall machine performance

and leads to errors if the process runs out of accessible memory. It is recommended that client side memory is not consumed in excess. This can be done by reducing the batch size.

A batch is complete when BatchSize number of rows have been processed or there are no more rows to send to the database.

- If BatchSize > 0 and the UseInternalTransaction bulk copy option is specified, each batch of the bulk copy operation occurs within a transaction. If the connection used to perform the bulk copy operation is already part of a transaction, an `InvalidOperationException` exception is raised.
- If BatchSize > 0 and the UseInternalTransaction option is not specified, rows are sent to the database in batches of size BatchSize, but no transaction-related action is taken.

The BatchSize property can be set at any time. If a bulk copy is already in progress, the current batch size is determined by the previous batch size. Subsequent batches use the new batch size.

If the BatchSize property is initially zero and changes while a `WriteToServer` operation is in progress, that operation loads the data as a single batch. Any subsequent `WriteToServer` operations on the same `OracleBulkCopy` instance use the new BatchSize.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopy Class](#)
- [OracleBulkCopy Members](#)
- ["OracleBulkCopyOptions Enumeration"](#)

BulkCopyOptions

This property specifies the `OracleBulkCopyOptions` enumeration value that determines the behavior of the bulk copy option.

Declaration

```
// C#  
public OracleBulkCopyOptions BulkCopyOptions {get; set;}
```

Property Value

The `OracleBulkCopyOptions` enumeration object that defines the behavior of the bulk copy operation.

Exceptions

`ArgumentNullException` - The bulk copy options set is null.

Remarks

The default value of this property is `OracleBulkCopyOptions.Default` value. This property can be used to change the bulk copy options between the batches of a bulk copy operation.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopy Class](#)
- [OracleBulkCopy Members](#)
- ["OracleBulkCopyOptions Enumeration"](#)

BulkCopyTimeout

This property specifies the number of seconds allowed for the bulk copy operation to complete before it is aborted.

Declaration

```
// C#  
public int BulkCopyTimeout {get; set;}
```

Property Value

An integer value for the number of seconds after which the bulk copy operation times out.

Exceptions

ArgumentOutOfRangeException - The timeout value is set to less than zero.

Remarks

The default value is 30 seconds.

If BatchSize>0, rows that were sent to the database in the previous batches remain committed. The rows that are processed in the current batch are not sent to the database. If BatchSize=0, no rows are sent to the database.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopy Class](#)
- [OracleBulkCopy Members](#)

ColumnMappings

This property specifies the column mappings between the data source and destination table.

Declaration

```
// C#  
public OracleBulkCopyColumnMappingCollection ColumnMappings {get;}
```

Property Value

The `OracleBulkCopyColumnMappingCollection` object that defines the column mapping between the source and destination table.

Remarks

The `ColumnMappings` collection is unnecessary if the data source and the destination table have the same number of columns, and the ordinal position of each source column matches the ordinal position of the corresponding destination column. However, if the column counts differ, or the ordinal positions are not consistent, the `ColumnMappings` collection must be used to ensure that data is copied into the correct columns.

During the execution of a bulk copy operation, this collection can be accessed, but it cannot be changed.

By default, this property specifies an empty collection of column mappings.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopy Class](#)
- [OracleBulkCopy Members](#)

Connection

This property specifies the `OracleConnection` object that the Oracle database uses to perform the bulk copy operation.

Declaration

```
// C#  
public OracleConnection Connection {get; }
```

Property Value

The `OracleConnection` object used for the bulk copy operations.

Remarks

This property gets the connection constructed by the `OracleBulkCopy`, if the `OracleBulkCopy` object is initialized using a connection string.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopy Class](#)
- [OracleBulkCopy Members](#)

DestinationPartitionName

This property specifies the database partition that the data is loaded into.

Declaration

```
// C#  
public string DestinationPartitionName {get; set;}
```

Property Value

A string value that identifies the destination partition name.

Remarks

If `DestinationPartitionName` is modified while a `WriteToServer` operation is running, the change does not affect the current operation. The new `DestinationPartitionName` value is used the next time a `WriteToServer` method is called.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopy Class](#)
- [OracleBulkCopy Members](#)

DestinationSchemaName

This property specifies the database schema that the data is loaded into.

Declaration

```
// C#  
public string DestinationSchemaName {get; set;}
```

Property Value

A string value that identifies the destination schema name.

Remarks

By default, this property is set to the schema or the user associated with the connection used by the `OracleBulkCopy` object. This default can be modified by setting this property to a different destination schema or user name.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopy Class](#)
- [OracleBulkCopy Members](#)

DestinationTableName

This property specifies the database table that the data is loaded into.

Declaration

```
// C#  
public string DestinationTableName {get; set;}
```

Property Value

A string value that identifies the destination table name.

Exceptions

`ArgumentNullException` - The destination table name set is null.

`ArgumentException` - The destination table name is empty.

Remarks

If `DestinationTableName` is modified while a `WriteToServer` operation is running, the change does not affect the current operation. The new `DestinationTableName` value is used the next time a `WriteToServer` method is called.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopy Class](#)
- [OracleBulkCopy Members](#)

NotifyAfter

This property defines the number of rows to be processed before a notification event is generated.

Declaration

```
// C#  
public int NotifyAfter {get; set;}
```

Property Value

An integer value that specifies the number of rows to be processed before the notification event is raised.

Exceptions

`ArgumentOutOfRangeException` - The property value is set to a number less than zero.

Remarks

The default value for this property is zero, to specify that no notifications events are to be generated.

This property can be retrieved in user interface components to display the progress of a bulk copy operation. The `NotifyAfter` property can be set at anytime, even during a bulk copy operation. The changes take effect for the next notification and any subsequent operations on the same instance. By ADO.NET design, this property is intended to measure the progress of a bulk copy operation, not when all rows have been processed. To be alerted when all rows have been processed, use `OracleBulkCopyOptions.NotifyAllRowProcessed`.

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopy Class](#)
- [OracleBulkCopy Members](#)

OracleBulkCopy Public Methods

OracleBulkCopy methods are listed in [Table 17-6](#).

Table 17-6 OracleBulkCopy Public Methods

Method	Description
Close	Closes the OracleBulkCopy instance
Dispose	Releases any resources or memory allocated by the object
WriteToServer	Copies rows to a destination table

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopy Class](#)
- [OracleBulkCopy Members](#)

Close

This method closes the OracleBulkCopy instance.

Declaration

```
// C#  
public void Close();
```

Exceptions

`InvalidOperationException` - The Close method was called from a `OracleRowsCopied` event.

Remarks

After the Close method is called on a OracleBulkCopy object, no other operation can succeed. Calls to the WriteToServer method throw an `InvalidOperationException`. The Close method

closes the connection if the connection was opened by the `OracleBulkCopy` object, that is, if the `OracleBulkCopy` object was created by a constructor that takes a connection string.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopy Class](#)
- [OracleBulkCopy Members](#)

Dispose

This method releases any resources or memory allocated by the object.

Declaration

```
// C#  
public void Dispose();
```

Implements

`IDisposable`

Remarks

After the `Dispose` method is called on the `OracleBulkCopy` object, no other operation can succeed. The connection is closed if the connection was opened by the `OracleBulkCopy` object, that is, if a constructor that takes a connection string created the `OracleBulkCopy` object.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopy Class](#)
- [OracleBulkCopy Members](#)

WriteToServer

`WriteToServer` copies rows to a destination table.

Overload List:

- [WriteToServer\(DataRow\[\]\)](#)

This method copies all rows from the supplied `DataRow` array to a destination table specified by the `DestinationTableName` property of the `OracleBulkCopy` object.

- [WriteToServer\(DataTable\)](#)

This method copies all rows in the supplied `DataTable` to a destination table specified by the `DestinationTableName` property of the `OracleBulkCopy` object.

- [WriteToServer\(IDataReader\)](#)

This method copies all rows in the supplied `IDataReader` to a destination table specified by the `DestinationTableName` property of the `OracleBulkCopy` object.

- [WriteToServer\(DataTable, DataRowState\)](#)

This method copies rows that match the supplied row state in the supplied `DataTable` to a destination table specified by the `DestinationTableName` property of the `OracleBulkCopy` object.

- [WriteToServer\(OracleRefCursor\)](#)

This method copies all rows from the specified `OracleRefCursor` to a destination table specified by the `DestinationTableName` property of the `OracleBulkCopy` object.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopy Class](#)
- [OracleBulkCopy Members](#)

WriteToServer(DataRow[])

This method copies all rows from the supplied `DataRow` array to a destination table specified by the `DestinationTableName` property of the `OracleBulkCopy` object.

Declaration

```
// C#  
public void WriteToServer(DataRow[] rows);
```

Parameters

- *rows*

An array of `DataRow` objects to be copied to the destination table.

Exceptions

`ArgumentNullException` - The *rows* parameter is null.

`InvalidOperationException` - The connection is not in an open state.

Remarks

The `ColumnMappings` collection maps from the `DataRow` columns to the destination database table.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopy Class](#)
- [OracleBulkCopy Members](#)

WriteToServer(DataTable)

This method copies all rows in the supplied DataTable to a destination table specified by the DestinationTableName property of the OracleBulkCopy object.

Declaration

```
// C#  
public void WriteToServer(DataTable table);
```

Parameters

- *table*

The source DataTable containing rows to be copied to the destination table.

Exceptions

ArgumentNullException - The *table* parameter is null.

InvalidOperationException - The connection is not in an open state.

Remarks

All rows in the DataTable are copied to the destination table except those that have been deleted.

The ColumnMappings collection maps from the DataTable columns to the destination database table.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopy Class](#)
- [OracleBulkCopy Members](#)

WriteToServer(IDataReader)

This method copies all rows in the supplied IDataReader to a destination table specified by the DestinationTableName property of the OracleBulkCopy object.

Declaration

```
// C#  
public void WriteToServer(IDataReader reader);
```


Parameters

- *reader*
A *IDataReader* instance containing rows to be copied to the destination table.

Exceptions

ArgumentNullException - The *reader* parameter is null.

InvalidOperationException - The connection is not in an open state.

Remarks

The bulk copy operation starts with the next available row of the data reader. Typically, the *reader* returned by a call to the *ExecuteReader* method is passed to the *WriteToServer* method so that the next row becomes the first row. To copy multiple result sets, the application must call *NextResult* on the *reader* and then call the *WriteToServer* method again.

This *WriteToServer* method changes the state of the reader as it calls *reader.Read* internally to get the source rows. Thus, at the end of the *WriteToServer* operation, the *reader* is at the end of the result set.

The *ColumnMappings* collection maps from the data reader columns to the destination database table.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopy Class](#)
- [OracleBulkCopy Members](#)

WriteToServer(DataTable, DataRowState)

This method copies rows that match the supplied row state in the supplied *DataTable* to a destination table specified by the *DestinationTableName* property of the *OracleBulkCopy* object.

Declaration

```
// C#  
public void WriteToServer(DataTable table, DataRowState rowState);
```

Parameters

- *table*
A *DataTable* containing rows to be copied to the destination table.
- *rowState*
The *DataRowState* enumeration value. Only rows matching the row state are copied to the destination.

Exceptions

ArgumentNullException - The *table* or *rowState* parameter is null.

InvalidOperationException - The connection is not in an open state.

Remarks

Only rows in the DataTable that are in the state indicated in the *rowState* argument and have not been deleted are copied to the destination table.

The ColumnMappings collection maps from the DataTable columns to the destination database table.

DataRowState.Deleted is not supported and the behavior would be that all the rows except the deleted ones are copied.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopy Class](#)
- [OracleBulkCopy Members](#)

WriteToServer(OracleRefCursor)

This method copies all rows from the specified OracleRefCursor to a destination table specified by the DestinationTableName property of the OracleBulkCopy object.

Declaration

```
// C#  
public void WriteToServer(OracleRefCursor refCursor);
```

Parameters

- *refCursor*
An OracleRefCursor object containing rows to be copied to the destination table.

Exceptions

ArgumentNullException - The *refCursor* parameter is null

InvalidOperationException - The connection is not in an open state.

Remarks

The ColumnMappings collection maps from the OracleRefCursor columns to the destination database table.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopy Class](#)
- [OracleBulkCopy Members](#)

OracleBulkCopy Events

OracleBulkCopy events are listed in [Table 17-7](#).

Table 17-7 OracleBulkCopy Events

Event	Description
OracleRowsCopied	Triggered every time the number of rows specified by the <code>OracleBulkCopy.NotifyAfter</code> property has been processed

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopy Class](#)
- [OracleBulkCopy Members](#)

OracleRowsCopied

This event is triggered every time the number of rows specified by the `OracleBulkCopy.NotifyAfter` property has been processed.

Declaration

```
// C#  
public event OracleRowsCopiedEventHandler OracleRowsCopied;
```

Exceptions

`InvalidOperationException` - The `Close` method is called inside this event.

Remarks

This event is raised when the number of rows specified by the `NotifyAfter` property has been processed. It does not imply that the rows have been sent to the database or committed.

To cancel the operation from this event, use the `Abort` property of `OracleRowsCopiedEventArgs` class.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopy Class](#)
- [OracleBulkCopy Members](#)
- ["NotifyAfter"](#)

OracleBulkCopyColumnMapping Class

The OracleBulkCopyColumnMapping class defines the mapping between a column in the data source and a column in the destination database table.

Class Inheritance

System.Object

System.OracleBulkCopyColumnMapping

Declaration

```
// C#
public sealed class OracleBulkCopyColumnMapping
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

Column mappings define the mapping between data source and the target table.

It is not necessary to specify column mappings for all the columns in the data source. If a ColumnMapping is not specified, then, by default, columns are mapped based on the ordinal position. This succeeds only if the source and destination table schema match. If there is a mismatch, an InvalidOperationException is thrown.

All the mappings in a mapping collection must be by name or ordinal position.

Note

Oracle Data Provider for .NET makes one or more round-trips to the database to determine the column name if the mapping is specified by ordinal position. To avoid this performance overhead, specify the mapping by column name.

Example

```
// C#
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMapping Members](#)
- [OracleBulkCopyColumnMapping Constructors](#)
- [OracleBulkCopyColumnMapping Methods](#)
- [OracleBulkCopyColumnMapping Properties](#)

OracleBulkCopyColumnMapping Members

OracleBulkCopyColumnMapping members are listed in the following tables.

OracleBulkCopyColumnMapping Constructors

The OracleBulkCopyColumnMapping constructors are listed in [Table 17-8](#).

Table 17-8 OracleBulkCopyColumnMapping Constructors

Constructor	Description
OracleBulkCopyColumnMapping Constructors	Instantiates new instances of the OracleBulkCopyColumnMapping class

OracleBulkCopyColumnMapping Methods

The OracleBulkCopyColumnMapping method is listed in [Table 17-9](#).

Table 17-9 OracleBulkCopyColumnMapping Method

Constructor	Description
CompareTo	Compares the current instance to the supplied object and returns an integer that represents their relative values

OracleBulkCopyColumnMapping Properties

The OracleBulkCopyColumnMapping properties are listed in [Table 17-10](#).

Table 17-10 OracleBulkCopyColumnMapping Properties

Property	Description
DestinationColumn	Specifies the column name of the destination table that is being mapped
DestinationOrdinal	Specifies the column ordinal value of the destination table that is being mapped
SourceColumn	Specifies the column name of the data source that is being mapped
SourceOrdinal	Specifies the column ordinal value of the data source that is being mapped

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleBulkCopyColumnMapping Class](#)

OracleBulkCopyColumnMapping Constructors

OracleBulkCopyColumnMapping constructors instantiates new instances of the OracleBulkCopyColumnMapping class.

Overload List:

- [OracleBulkCopyColumnMapping\(\)](#)
This constructor instantiates a new instance of the OracleBulkCopyColumnMapping class
- [OracleBulkCopyColumnMapping\(int, int\)](#)
This constructor instantiates a new instance of the OracleBulkCopyColumnMapping class using the provided source column ordinal and destination column ordinal.
- [OracleBulkCopyColumnMapping\(int, string\)](#)
This constructor instantiates a new instance of the OracleBulkCopyColumnMapping class using the provided source column ordinal and destination column name.
- [OracleBulkCopyColumnMapping\(string, int\)](#)
This constructor instantiates a new instance of the OracleBulkCopyColumnMapping class using the provided source column name and destination column ordinal.
- [OracleBulkCopyColumnMapping\(string, string\)](#)
This constructor instantiates a new instance of the OracleBulkCopyColumnMapping class using the provided source column name and destination column name.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMapping Class](#)
- [OracleBulkCopyColumnMapping Members](#)

OracleBulkCopyColumnMapping()

This constructor instantiates a new instance of the `OracleBulkCopyColumnMapping` class.

Declaration

```
// C#  
public OracleBulkCopyColumnMapping();
```

Remarks

Applications that use this constructor must define the source for the mapping using the `SourceColumn` or `SourceOrdinal` property, and must define the destination for the mapping using the `DestinationColumn` or `DestinationOrdinal` property.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMapping Class](#)
- [OracleBulkCopyColumnMapping Members](#)

OracleBulkCopyColumnMapping(int, int)

This constructor instantiates a new instance of the `OracleBulkCopyColumnMapping` class using the provided source and destination column ordinal positions.

Declaration

```
// C#  
public OracleBulkCopyColumnMapping(int sourceColumnOrdinal,  
    int destinationOrdinal);
```

Parameters

- *sourceColumnOrdinal*
The ordinal position of the source column within the data source.
- *destinationOrdinal*
The ordinal position of the destination column within the destination table.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMapping Class](#)
- [OracleBulkCopyColumnMapping Members](#)

OracleBulkCopyColumnMapping(int, string)

This constructor instantiates a new instance of the `OracleBulkCopyColumnMapping` class using the provided source column ordinal and destination column name.

Declaration

```
// C#  
public OracleBulkCopyColumnMapping(int sourceColumnOrdinal,  
    string destinationColumn);
```

Parameters

- *sourceColumnOrdinal*
The ordinal position of the source column within the data source.
- *destinationColumn*
The name of the destination column within the destination table.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMapping Class](#)
- [OracleBulkCopyColumnMapping Members](#)

OracleBulkCopyColumnMapping(string, int)

This constructor instantiates a new instance of the `OracleBulkCopyColumnMapping` class using the provided source column name and destination column ordinal.

Declaration

```
// C#  
public OracleBulkCopyColumnMapping(string sourceColumn, int destinationOrdinal);
```

Parameters

- *sourceColumn*
The name of the source column within the data source.
- *destinationOrdinal*
The ordinal position of the destination column within the destination table.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMapping Class](#)
- [OracleBulkCopyColumnMapping Members](#)

OracleBulkCopyColumnMapping(string, string)

This constructor instantiates a new instance of the `OracleBulkCopyColumnMapping` class using the provided source and destination column names.

Declaration

```
// C#
public OracleBulkCopyColumnMapping(string sourceColumn, string destinationColumn);
```

Parameters

- *sourceColumn*
The name of the source column within the data source.
- *destinationColumn*
The name of the destination column within the destination table.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMapping Class](#)
- [OracleBulkCopyColumnMapping Members](#)

OracleBulkCopyColumnMapping Methods

The `OracleBulkCopyColumnMapping` method is listed in [Table 17-11](#).

Table 17-11 OracleBulkCopyColumnMapping Method

Property	Description
CompareTo	Compares the current instance to the supplied object and returns an integer that represents their relative values

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMapping Class](#)
- [OracleBulkCopyColumnMapping Members](#)

CompareTo

This method compares the current instance to the supplied object and returns an integer that represents their relative values.

Declaration

```
// C#
public int CompareTo(object obj);
```

Parameters

obj - The supplied instance.

Return Value

Less than zero: if the value of the current instance is less than obj.
Zero: if the value of the current instance is equal to obj.
Greater than zero: if the value of the current instance is greater than obj.

Implements

IComparable

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMapping Class](#)
- [OracleBulkCopyColumnMapping Members](#)

OracleBulkCopyColumnMapping Properties

The OracleBulkCopyColumnMapping properties are listed in [Table 17-12](#).

Table 17-12 OracleBulkCopyColumnMapping Properties

Property	Description
DestinationColumn	Specifies the column name of the destination table that is being mapped
DestinationOrdinal	Specifies the column ordinal value of the destination table that is being mapped

Table 17-12 (Cont.) OracleBulkCopyColumnMapping Properties

Property	Description
SourceColumn	Specifies the column name of the data source that is being mapped
SourceOrdinal	Specifies the column ordinal value of the data source that is being mapped

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMapping Class](#)
- [OracleBulkCopyColumnMapping Members](#)

DestinationColumn

This property specifies the column name of the destination table that is being mapped.

Declaration

```
// C#  
public string DestinationColumn {get; set;}
```

Property Value

A string value that represents the destination column name of the mapping.

Remarks

The DestinationColumn and DestinationOrdinal properties are mutually exclusive. The last value set takes precedence.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMapping Class](#)
- [OracleBulkCopyColumnMapping Members](#)

DestinationOrdinal

This property specifies the column ordinal value of the destination table that is being mapped.

Declaration

```
// C#  
public int DestinationOrdinal {get; set;}
```

Property Value

An integer value that represents the destination column ordinal of the mapping.

Exceptions

`IndexOutOfRangeException` - The destination ordinal is invalid.

Remarks

The `DestinationOrdinal` and `DestinationColumn` properties are mutually exclusive. The last value set takes precedence.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMapping Class](#)
- [OracleBulkCopyColumnMapping Members](#)

SourceColumn

This property specifies the column name of the data source that is being mapped.

Declaration

```
// C#  
public string SourceColumn {get; set;}
```

Property Value

A string value that represents the source column name of the mapping.

Remarks

The `SourceColumn` and `SourceOrdinal` properties are mutually exclusive. The last value set takes precedence.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMapping Class](#)
- [OracleBulkCopyColumnMapping Members](#)

SourceOrdinal

This property specifies the column ordinal value of the data source that is being mapped.

Declaration

```
// C#
public int SourceOrdinal {get; set;}
```

Property Value

An integer value that represents the source column ordinal of the mapping.

Exceptions

IndexOutOfRangeException - The source ordinal is invalid.

Remarks

The SourceOrdinal and SourceColumn properties are mutually exclusive. The last value set takes precedence.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMapping Class](#)
- [OracleBulkCopyColumnMapping Members](#)

OracleBulkCopyColumnMappingCollection Class

The OracleBulkCopyColumnMappingCollection class represents a collection of OracleBulkCopyColumnMapping objects that are used to map columns in the data source to columns in a destination table.

Class Inheritance

```
System.Object
    System.CollectionBase
        System.OracleBulkCopyColumnMappingCollection
```

Declaration

```
// C#
public sealed class OracleBulkCopyColumnMappingCollection : CollectionBase
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

Column mappings define the mapping between data source and the target table.

It is not necessary to specify column mappings for all the columns in the data source. If a ColumnMapping is not specified, then, by default, columns are mapped based on the ordinal position. This succeeds only if the source and destination table schema match. If there is a mismatch, an `InvalidOperationException` is thrown.

All the mappings in a mapping collection must be by name or ordinal position.

Note

Oracle Data Provider for .NET makes one or more round-trips to the database to determine the column name if the mapping is specified by ordinal position. To avoid this performance overhead, specify the mapping by column name.

Example

```
// C#
```

See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- [OracleBulkCopyColumnMappingCollection Class](#)
- [OracleBulkCopyColumnMappingCollection Members](#)
- [OracleBulkCopyColumnMappingCollection Properties](#)
- [OracleBulkCopyColumnMappingCollection Public Methods](#)

OracleBulkCopyColumnMappingCollection Members

OracleBulkCopyColumnMappingCollection members are listed in the following tables.

OracleBulkCopyColumnMappingCollection Properties

The OracleBulkCopyColumnMappingCollection properties are listed in [Table 17-13](#).

Table 17-13 OracleBulkCopyColumnMappingCollection Properties

Property	Description
Item[index]	Gets or sets the OracleBulkCopyColumnMappingCollection object at the specified index

OracleBulkCopyColumnMappingCollection Public Methods

The OracleBulkCopyColumnMappingCollection public methods are listed in [Table 17-14](#).

Table 17-14 OracleBulkCopyColumnMappingCollection Public Methods

Public Method	Description
Add	Adds objects to the collection
Clear	Clears the contents of the collection
Contains	Returns a value indicating whether or not a specified OracleBulkCopyColumnMapping object exists in the collection
CopyTo	Copies the elements of the OracleBulkCopyColumnMappingCollection to an array of OracleBulkCopyColumnMapping items, starting at a specified index
IndexOf	Returns the index of the specified OracleBulkCopyColumnMapping object
Insert	Inserts a new OracleBulkCopyColumnMapping object in the collection, at the index specified.
Remove	Removes the specified OracleBulkCopyColumnMapping element from the OracleBulkCopyColumnMappingCollection.
RemoveAt	Removes the mapping from the collection at the specified index.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMappingCollection Class](#)

OracleBulkCopyColumnMappingCollection Properties

The OracleBulkCopyColumnMappingCollection properties are listed in [Table 17-15](#).

Table 17-15 OracleBulkCopyColumnMappingCollection Properties

Property	Description
Item[index]	Gets or sets the OracleBulkCopyColumnMappingCollection object at the specified index

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMappingCollection Class](#)
- [OracleBulkCopyColumnMappingCollection Members](#)

Item[index]

This property gets or sets the OracleBulkCopyColumnMapping object at the specified index.

Declaration

```
// C#  
public OracleBulkCopyColumnMapping this[int index] {get;set;}
```

Parameters

- *index*

The zero-based index of the OracleBulkCopyColumnMapping being set or retrieved.

Property Value

An OracleBulkCopyColumnMapping object at the specified index.

Exceptions

IndexOutOfRangeException - The specified index does not exist.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMappingCollection Class](#)
- [OracleBulkCopyColumnMappingCollection Members](#)

OracleBulkCopyColumnMappingCollection Public Methods

The OracleBulkCopyColumnMappingCollection public methods are listed in [Table 17-16](#).

Table 17-16 OracleBulkCopyColumnMappingCollection Public Methods

Public Method	Description
Add	Adds objects to the collection
Clear	Clears the contents of the collection
Contains	Returns a value indicating whether or not a specified OracleBulkCopyColumnMapping object exists in the collection

Table 17-16 (Cont.) OracleBulkCopyColumnMappingCollection Public Methods

Public Method	Description
CopyTo	Copies the elements of the OracleBulkCopyColumnMappingCollection to an array of OracleBulkCopyColumnMapping items, starting at a specified index
IndexOf	Returns the index of the specified OracleBulkCopyColumnMapping object
Insert	Inserts a new OracleBulkCopyColumnMapping object in the collection, at the index specified.
Remove	Removes the specified OracleBulkCopyColumnMapping element from the OracleBulkCopyColumnMappingCollection.
RemoveAt	Removes the mapping from the collection at the specified index.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMappingCollection Class](#)
- [OracleBulkCopyColumnMappingCollection Members](#)

Add

Add methods add objects to the collection.

Overload List:

- [Add\(OracleBulkCopyColumnMapping\)](#)
This method adds the supplied OracleBulkCopyColumnMapping object to the collection.
- [Add\(int, int\)](#)
This method creates and adds an OracleBulkCopyColumnMapping object to the collection using the supplied source and destination column ordinal positions.
- [Add\(int, string\)](#)
This method creates and adds an OracleBulkCopyColumnMapping object to the collection using the supplied source column ordinal and destination column name.
- [Add\(string, int\)](#)
This method creates and adds an OracleBulkCopyColumnMapping object to the collection using the supplied source column name and destination column ordinal.
- [Add\(string, string\)](#)
This method creates and adds an OracleBulkCopyColumnMapping object to the collection using the supplied source and destination column names.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMappingCollection Class](#)
- [OracleBulkCopyColumnMappingCollection Members](#)

Add(OracleBulkCopyColumnMapping)

This method adds the supplied `OracleBulkCopyColumnMapping` object to the collection.

Declaration

```
// C#
public OracleBulkCopyColumnMapping Add(OracleBulkCopyColumnMapping
    bulkCopyColumnMapping);
```

Parameters

- *bulkCopyColumnMapping*

The `OracleBulkCopyColumnMapping` object that describes the mapping to be added to the collection.

Exceptions

`InvalidOperationException` - The bulk copy operation is in progress.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMappingCollection Class](#)
- [OracleBulkCopyColumnMappingCollection Members](#)

Add(int, int)

This method creates and adds an `OracleBulkCopyColumnMapping` object to the collection using the supplied source and destination column ordinal positions.

Declaration

```
// C#
public OracleBulkCopyColumnMapping Add(int sourceColumnIndex,
    int destinationColumnIndex);
```

Parameters

- *sourceColumnIndex*
The ordinal position of the source column within the data source.
- *destinationColumnIndex*

The ordinal position of the destination column within the destination table.

Exceptions

InvalidOperationException - The bulk copy operation is in progress.

Return Value

The newly created OracleBulkCopyColumnMapping object that was added to the collection.

Remarks

It is not necessary to specify column mappings for all the columns in the data source. If a ColumnMapping is not specified, then, by default, columns are mapped based on the ordinal position. This succeeds only if the source and destination table schema match. If there is a mismatch, an InvalidOperationException is thrown.

All the mappings in a mapping collection must be by name or ordinal position.

Note

Oracle Data Provider for .NET makes one or more round-trips to the database to determine the column name if the mapping is specified by ordinal position. To avoid this performance overhead, specify the mapping by column name.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMappingCollection Class](#)
- [OracleBulkCopyColumnMappingCollection Members](#)

Add(int, string)

This method creates and adds an OracleBulkCopyColumnMapping object to the collection using the supplied source column ordinal and destination column name.

Declaration

```
// C#  
public OracleBulkCopyColumnMapping Add(int sourceColumnIndex,  
    string destinationColumn);
```

Parameters

- *sourceColumnIndex*
The ordinal position of the source column within the data source.
- *destinationColumn*
The name of the destination column within the destination table.

Exceptions

InvalidOperationException - The bulk copy operation is in progress.

Return Value

The newly created OracleBulkCopyColumnMapping object that was added to the collection.

Remarks

It is not necessary to specify column mappings for all the columns in the data source. If a ColumnMapping is not specified, then, by default, columns are mapped based on the ordinal position. This succeeds only if the source and destination table schema match. If there is a mismatch, an InvalidOperationException is thrown.

All the mappings in a mapping collection must be by name or ordinal position.

Note

Oracle Data Provider for .NET makes one or more round trips to the database to determine the column names if the mapping is specified by ordinal resulting in a performance overhead. Therefore, it is recommended to specify the mapping by column names.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMappingCollection Class](#)
- [OracleBulkCopyColumnMappingCollection Members](#)

Add(string, int)

This method creates and adds an OracleBulkCopyColumnMapping object to the collection using the supplied source column name and destination column ordinal.

Declaration

```
// C#  
public OracleBulkCopyColumnMapping Add(string sourceColumn,  
    int destinationColumnIndex);
```

Parameters

- *sourceColumn*
The name of the source column within the data source.
- *destinationColumnIndex*
The ordinal position of the destination column within the destination table.

Return Value

The newly created `OracleBulkCopyColumnMapping` object that was added to the collection.

Exceptions

`InvalidOperationException` - The bulk copy operation is in progress.

Remarks

It is not necessary to specify column mappings for all the columns in the data source. If a `ColumnMapping` is not specified, then, by default, columns are mapped based on the ordinal position. This succeeds only if the source and destination table schema match. If there is a mismatch, an `InvalidOperationException` is thrown.

All the mappings in a mapping collection must be by name or ordinal position.

Note

Oracle Data Provider for .NET makes one or more round trips to the database to determine the column names if the mapping is specified by ordinal resulting in a performance overhead. Therefore, it is recommended to specify the mapping by column names.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMappingCollection Class](#)
- [OracleBulkCopyColumnMappingCollection Members](#)

Add(string, string)

This method creates and adds an `OracleBulkCopyColumnMapping` object to the collection using the supplied source and destination column names.

Declaration

```
// C#  
public OracleBulkCopyColumnMapping Add(string sourceColumn,  
    string destinationColumn);
```

Parameters

- *sourceColumn*
The name of the source column within the data source.
- *destinationColumn*
The name of the destination column within the destination table.

Exceptions

InvalidOperationException - The bulk copy operation is in progress.

Return Value

The newly created OracleBulkCopyColumnMapping object that was added to the collection.

Remarks

It is not necessary to specify column mappings for all the columns in the data source. If a ColumnMapping is not specified, then, by default, columns are mapped based on the ordinal position. This succeeds only if the source and destination table schema match. If there is a mismatch, an InvalidOperationException is thrown.

All the mappings in a mapping collection must be by name or ordinal position.

Note

Oracle Data Provider for .NET makes one or more round-trips to the database to determine the column name if the mapping is specified by ordinal position. To avoid this performance overhead, specify the mapping by column name.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMappingCollection Class](#)
- [OracleBulkCopyColumnMappingCollection Members](#)

Clear

This method clears the contents of the collection.

Declaration

```
// C#  
public void Clear();
```

Exceptions

InvalidOperationException - The bulk copy operation is in progress.

Remarks

The Clear method is most commonly used when an application uses a single OracleBulkCopy instance to process more than one bulk copy operation. If column mappings are created for one bulk copy operation, the OracleBulkCopyColumnMappingCollection must be cleared after the WriteToServer method invocation and before the next bulk copy is processed.

It is usually more efficient to perform several bulk copies using the same OracleBulkCopy instance than to use a separate OracleBulkCopy for each operation.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMappingCollection Class](#)
- [OracleBulkCopyColumnMappingCollection Members](#)

Contains

This method returns a value indicating whether or not a specified OracleBulkCopyColumnMapping object exists in the collection.

Declaration

```
// C#  
public bool Contains(OracleBulkCopyColumnMapping value);
```

Parameters

- *value*
A valid OracleBulkCopyColumnMapping object.

Return Value

Returns true if the specified mapping exists in the collection; otherwise, returns false.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMappingCollection Class](#)
- [OracleBulkCopyColumnMappingCollection Members](#)

CopyTo

This method copies the elements of the OracleBulkCopyColumnMappingCollection to an array of OracleBulkCopyColumnMapping items, starting at a specified index.

Declaration

```
// C#  
public void CopyTo(OracleBulkCopyColumnMapping[] array, int index);
```

Parameters

- *array*
The one-dimensional OracleBulkCopyColumnMapping array that is the destination for the elements copied from the OracleBulkCopyColumnMappingCollection object. The array must have zero-based indexing.
- *index*

The zero-based array index at which copying begins.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMappingCollection Class](#)
- [OracleBulkCopyColumnMappingCollection Members](#)

IndexOf

This method returns the index of the specified `OracleBulkCopyColumnMapping` object.

Declaration

```
// C#  
public int IndexOf(OracleBulkCopyColumnMapping value);
```

Parameters

- *value*
The `OracleBulkCopyColumnMapping` object that is being returned.

Return Value

The zero-based index of the column mapping or -1 if the column mapping is not found in the collection.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMappingCollection Class](#)
- [OracleBulkCopyColumnMappingCollection Members](#)

Insert

This method inserts a new `OracleBulkCopyColumnMapping` object in the collection, at the index specified.

Declaration

```
// C#  
public void Insert(int index, OracleBulkCopyColumnMapping value);
```

Parameters

- *index*
The integer value of the location within the `OracleBulkCopyColumnMappingCollection` at which the new `OracleBulkCopyColumnMapping` is inserted.

- *value*

The `OracleBulkCopyColumnMapping` object to be inserted in the collection.

Exceptions

`InvalidOperationException` - The bulk copy operation is in progress.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMappingCollection Class](#)
- [OracleBulkCopyColumnMappingCollection Members](#)

Remove

This method removes the specified `OracleBulkCopyColumnMapping` element from the `OracleBulkCopyColumnMappingCollection`.

Declaration

```
// C#  
public void Remove(OracleBulkCopyColumnMapping value);
```

Parameters

- *value*

The `OracleBulkCopyColumnMapping` object to be removed from the collection.

Exceptions

`InvalidOperationException` - The bulk copy operation is in progress.

Remarks

The `Remove` method is most commonly used when a single `OracleBulkCopy` instance processes more than one bulk copy operation. If column mappings are created for one bulk copy operation, mappings that no longer apply must be removed after the `WriteToServer` method invocation and before mappings are defined for the next bulk copy. The `Clear` method can clear the entire collection, and the `Remove` and the `RemoveAt` methods can remove mappings individually.

It is usually more efficient to perform several bulk copies using the same `OracleBulkCopy` instance than to use a separate `OracleBulkCopy` for each operation.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMappingCollection Class](#)
- [OracleBulkCopyColumnMappingCollection Members](#)

RemoveAt

This method removes the mapping from the collection at the specified index.

Declaration

```
// C#  
public void RemoveAt(int index);
```

Parameters

- *index*
The zero-based index of the `OracleBulkCopyColumnMapping` object to be removed from the collection.

Exceptions

`InvalidOperationException` - The bulk copy operation is in progress.

Remarks

The `RemoveAt` method is most commonly used when a single `OracleBulkCopy` instance is used to process more than one bulk copy operation. If column mappings are created for one bulk copy operation, mappings that no longer apply must be removed after the `WriteToServer` method invocation and before the mappings for the next bulk copy are defined. The `Clear` method can clear the entire collection, and the `Remove` and the `RemoveAt` methods can remove mappings individually.

It is usually more efficient to perform several bulk copies using the same `OracleBulkCopy` instance than to use a separate `OracleBulkCopy` for each operation.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleBulkCopyColumnMappingCollection Class](#)
- [OracleBulkCopyColumnMappingCollection Members](#)

OracleBulkCopyOptions Enumeration

The `OracleBulkCopyOptions` enumeration specifies the values that can be combined with an instance of the `OracleBulkCopy` class and used as options to determine its behavior and the behavior of the `WriteToServer` methods for that instance.

[Table 17-17](#) lists all the `OracleBulkCopyOptions` enumeration values with a description of each enumerated value.

Table 17-17 OracleBulkCopyOptions Enumeration Members

Member Name	Description
Default	Indicates that the default value for all options are to be used

Table 17-17 (Cont.) OracleBulkCopyOptions Enumeration Members

Member Name	Description
EnforceIndexChecks	Indicates when a direct path load results in an index-related error, the load is aborted. No rows are loaded, and indexes remain unchanged. When this option is not specified and a direct path load results in an index becoming unusable, the rows are loaded and the index is left in an unusable state. Oracle database primary key and unique constraints are enforced using unique indexes, which means the database automatically creates unique indexes on columns specified as primary key or unique. Therefore, this option only affects primary key and unique constraints. This option behavior is similar to the NO_INDEX_ERRORS parameter in SQL *LOADER. This member is supported with managed ODP.NET and ODP.NET Core 23.8 and higher. Note While using ODP.NET bulk copy, the check and not null constraints are automatically enforced, whereas the foreign key constraint is never enforced.
NotifyAllRowsProcessed	Indicates when bulk copy has processed all the rows by sending a notification. This member can be used in conjunction with OracleBulkCopy.NotifyAfter or by itself. NotifyAfter sends notifications after a specific number of rows have been bulk copied. It only sends an alert when the entire bulk copy operation is completed if the NotifyAfter value is a multiple of the total rows copied. For example, let's say you set NotifyAfter=5 without enabling NotifyAllRowsProcessed. When you bulk copy 15 rows, you will receive three alerts: once after 5 rows, once after 10, and once after 15. If you bulk 12 rows instead, then you would receive only two alerts: once after 5 rows and once after 10 rows. If you had also enabled NotifyAllRowsProcessed in both scenarios, then you would receive three alerts in both instances and always after the bulk copy operation completes processing the 15 rows. NotifyAllRowsProcessed will send a notification when all the rows are processed regardless of NotifyAfter's value. By design, NotifyAfter is intended to measure bulk copy progress. NotifyAllRowsProcessed is intended to indicate when bulk copy has processed all the rows. This member is supported with managed ODP.NET and ODP.NET Core only starting with versions 23.5, 21.15, and 19.24.

Table 17-17 (Cont.) OracleBulkCopyOptions Enumeration Members

Member Name	Description
UseInternalTransaction	Indicates that each batch of the bulk copy operation occurs within a transaction. If the connection used to perform the bulk copy operation is already part of a transaction, an <code>InvalidOperationException</code> exception is raised. If this member is not specified, <code>BatchSize</code> number of rows are sent to the database, without any transaction-related activity.

Note

All bulk copy operations are agnostic of any local or distributed transaction created by the application.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Sample Code: NotifyAllRowsProcessed

```
// Set up the bulk copy object so that it will notify the app when all rows have been
// copied.
using (OracleBulkCopy bulkCopy = new OracleBulkCopy(destinationConnection,
OracleBulkCopyOption.NotifyAllRowsProcessed))
{
    bulkCopy.DestinationTableName = "BLOGS";

    try
    {
        // Write rows from the source to the destination.
        bulkCopy.NotifyAfter = 5;
        // OnSqlRowsCopied is the event handler delegate whenever NotifyAfter or
        // NotifyAllRowsProcessed sends a notification. OnSqlRowsCopied's implementation is
        // not included in this code sample.
        bulkCopy.OracleRowsCopied += new OracleRowsCopiedEventHandler(OnSqlRowsCopied);
        bulkCopy.WriteToServer(reader);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}
```

```
    }  
    finally  
    {  
        // Close the OracleDataReader. The OracleBulkCopy object is automatically closed at  
        // the end of the using block.  
        reader.Close();  
    }  
}
```

 See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- OracleBulkCopy ["BulkCopyOptions"](#)

OracleRowsCopiedEventHandler Delegate

The OracleRowsCopiedEventHandler delegate represents the method that handles the OracleRowsCopied event of an OracleBulkCopy object.

Declaration

```
// C#  
public delegate void OracleRowsCopiedEventHandler (object sender,  
    OracleRowsCopiedEventArgs eventArgs);
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Parameters

- *sender*
The source of the event.
- *eventArgs*
The OracleRowsCopiedEventArgs object that contains the event data.

Remarks

Event callbacks can be registered through this event delegate for applications that wish to be notified every time the number of rows specified by the OracleBulkCopy.NotifyAfter property has been processed.

If the event handler calls the `OracleBulkCopy.Close` method, an exception is generated, and the `OracleBulkCopy` object state does not change.

The event handler can also set the `OracleRowsCopiedEventArgs.Abort` property to `true` to indicate that the bulk copy operation must be aborted. If the bulk copy operation is part of an external transaction, an exception is generated and the transaction is not rolled back. The application is responsible for either committing or rolling back the external transaction.

If there is no external transaction, the internal transaction for the current batch of rows is automatically rolled back. However the previous batches of imported rows are unaffected, as their transactions have already been committed.

① See Also

- "[Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)"
- "[OracleRowsCopied](#)"
- "[NotifyAfter](#)"

OracleRowsCopiedEventArgs Class

The `OracleRowsCopiedEventArgs` class represents the set of arguments passed as part of event data for the `OracleRowsCopied` event.

Class Inheritance

System.Object
 System.EventArgs
 System.OracleRowsCopiedEventArgs

Declaration

```
// C#  
public class OracleRowsCopiedEventArgs : EventArgs
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	See System Requirements	-
.NET (Core)	-	-	See System Requirements

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

Each time the number of rows represented by the `OracleBulkCopy.NotifyAfter` property is processed, the `OracleBulkCopy.RowsCopied` event is raised, providing an `OracleRowsCopiedEventArgs` object that stores the event data.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRowsCopiedEventArgs Members](#)
- [OracleRowsCopiedEventArgs Constructors](#)
- [OracleRowsCopiedEventArgs Properties](#)

OracleRowsCopiedEventArgs Members

`OracleRowsCopiedEventArgs` members are listed in the following tables.

OracleRowsCopiedEventArgs Constructors

`OracleRowsCopiedEventArgs` constructors are listed in [Table 17-18](#).

Table 17-18 OracleRowsCopiedEventArgs Constructors

Constructor	Description
OracleRowsCopiedEventArgs Constructors .	<code>OracleRowsCopiedEventArgs</code> creates new instances of the <code>OracleRowsCopiedEventArgs</code> class

OracleRowsCopiedEventArgs Properties

`OracleRowsCopiedEventArgs` properties are listed in [Table 17-19](#).

Table 17-19 OracleRowsCopiedEventArgs Properties

Property	Description
Abort	Retrieves or sets a value that indicates whether or not the bulk copy operation is aborted
RowsCopied	Retrieves a value that represents the number of rows copied during the current bulk copy operation

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRowsCopiedEventArgs Class](#)

OracleRowsCopiedEventArgs Constructors

OracleRowsCopiedEventArgs creates new instances of the OracleRowsCopiedEventArgs class.

Overload List:

- [OracleRowsCopiedEventArgs\(long\)](#)

This constructor creates a new instance of the OracleRowsCopiedEventArgs object.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRowsCopiedEventArgs Class](#)
- [OracleRowsCopiedEventArgs Members](#)

OracleRowsCopiedEventArgs(long)

This constructor creates a new instance of the OracleRowsCopiedEventArgs object.

Declaration

```
// C#  
public OracleRowsCopiedEventArgs(long rowsCopied);
```

Parameters

- *rowsCopied*
An Int64 value that indicates the number of rows copied during the current bulk copy operation.

Remarks

The value in the *rowsCopied* parameter is reset by each call to a WriteToServer method.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRowsCopiedEventArgs Class](#)
- [OracleRowsCopiedEventArgs Members](#)

OracleRowsCopiedEventArgs Properties

OracleRowsCopiedEventArgs properties are listed in [Table 17-20](#).

Table 17-20 OracleRowsCopiedEventArgs Properties

Property	Description
Abort	Retrieves or sets a value that indicates whether or not the bulk copy operation is aborted
RowsCopied	Retrieves a value that represents the number of rows copied during the current bulk copy operation

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRowsCopiedEventArgs Class](#)
- [OracleRowsCopiedEventArgs Members](#)

Abort

This property retrieves or sets a value that indicates whether or not the bulk copy operation is aborted.

Declaration

```
// C#  
public bool Abort{get; set;}
```

Property Value

Returns true if the bulk copy operation is to be aborted; otherwise, returns false.

Remarks

Set the Abort property to true to cancel the bulk copy operation.

If the Close method is called from OracleRowsCopied, an exception is generated, and the OracleBulkCopy object state does not change.

If the application does not create a transaction, the internal transaction corresponding to the current batch is automatically rolled back. However, changes related to previous batches within the bulk copy operation are retained, because the transactions in those batches are committed. This case is applicable only when UseInternalTransaction bulk copy option is chosen.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRowsCopiedEventArgs Class](#)
- [OracleRowsCopiedEventArgs Members](#)

RowsCopied

This property retrieves a value that represents the number of rows copied during the current bulk copy operation.

Declaration

```
// C#  
public long RowsCopied {get;}
```

Property Value

An Int64 value that returns the number of rows copied.

Remarks

The value in the RowsCopied property is reset by each call to a WriteToServer method.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleRowsCopiedEventArgs Class](#)
- [OracleRowsCopiedEventArgs Members](#)

Oracle Data Provider for .NET OpenTelemetry and Metrics Classes

This chapter describes ODP.NET OpenTelemetry classes and objects.

OracleDataProviderInstrumentationOptions Class is available through ODP.NET Core and managed ODP.NET. AddOracleDataProviderInstrumentation extension method is available through Oracle.ManagedDataAccess.OpenTelemetry NuGet package, which can be used with ODP.NET Core and managed ODP.NET.

```
public static TracerProviderBuilder AddOracleDataProviderInstrumentation(this
TracerProviderBuilder
builder, Action<OracleDataProviderInstrumentationOptions>
configureInstrumentationOptions = null)
```

Client applications call this extension method to enable OpenTelemetry instrumentation in ODP.NET.

This method accepts options to enable or disable different instrumentation types.

- [OracleDataProviderInstrumentationOptions Class](#)
- [AddOracleDataProviderInstrumentation](#) extension method

OracleDataProviderInstrumentationOptions Class

OracleDataProviderInstrumentationOptions class is available through the Oracle.ManagedDataAccess and Oracle.ManagedDataAccess.Core NuGet package.

This option class provides options passed to the AddOracleDataProviderInstrumentation extension method for enabling or disabling ODP.NET instrumentation types and tags.

Declaration

```
// C#
public class OracleDataProviderInstrumentationOptions
```

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

① See Also

- [Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)
- [OracleDataProviderInstrumentationOptions Properties](#)

OracleDataProviderInstrumentationOptions Properties

OracleDataProviderInstrumentationOptions properties are listed in [Table 18-1](#).

Table 18-1 OracleDataProviderInstrumentationOptions Properties

Method	Description
AddDBInfoToDisplayName	Indicates whether or not to record Host, Port and database name with Displayname
EnableConnectionLevelAttributes	Indicates whether or not to instrument connection level attributes, such as peer.service, db.name, password less connection string, db.user and so on
EnableDBRoundTripTracing	Allows creating child spans that encapsulate the database roundtrips for parent spans in public APIs like <code>OracleCommand.ExecuteNonQuery()</code> , <code>OracleDataReader.Read()</code> , and <code>OracleDataAdapter.Fill</code>
EnableOpenCloseTracing	Indicates whether to record database soft and hard connection/disconnection instrumentation or not
EnableSqlIdTracing	Enables SQL identifier instrumentation of each executed SQL statement
InstrumentOracleDataReaderRead	Indicates whether or not to instrument <code>OracleDataReader.Read</code> method calls
RecordException	Indicates whether or not to record exception details in instrumentation
RequireApplicationRootSpanCreation	Controls whether ODP.NET instruments only those operations that are associated with root spans created by the application or not
SetDbStatementForStoredProcedure	Indicates whether or not to instrument stored procedure and function names when the command type is for a stored procedure (<code>CommandType.StoredProcedure</code>)
SetDbStatementForText	Indicates whether or not to instrument SQL queries when command type is of Text type (<code>CommandType.Text</code>)

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataProviderInstrumentationOptions Class](#)

AddDBInfoToDisplayName

This property indicates whether or not to add Host, Port and database name to Displayname.
Example: Displayname <DB Host/IP>:<DB Port>:<DB Name>

Declaration

```
// C#  
public bool AddDBInfoToDisplayName { get; set; }
```

Return Value

A bool.

Remarks

Default value is false.

Note

If `EnableConnectionLevelAttributes` is set to false, then Host, Port, and database name information will not be added to `DisplayName`.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataProviderInstrumentationOptions Class](#)
- [OracleDataProviderInstrumentationOptions Properties](#)

EnableConnectionLevelAttributes

This property indicates whether or not to instrument connection level attributes, such as `peer.service`, `db.name`, password less connection string, `db.user` and so on.

Declaration

```
// C#  
public bool EnableConnectionLevelAttributes { get; set; }
```

Return Value

A bool.

Remarks

Default value is false.

When enabled, the `db.odp.connection.id` OpenTelemetry attribute publishes a unique identifier for each connection. This attribute can be used to diagnose and trace a specific connection's behavior. The identifier can be prefixed with an ODP.NET connection ID when the `ConnectionIdPrefix` property in `OracleConnection` or `OracleCommand` is set to an ID value.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataProviderInstrumentationOptions Class](#)
- [OracleDataProviderInstrumentationOptions Properties](#)

EnableDBRoundTripTracing

This property allows creating child spans that encapsulate the database roundtrips for parent spans in public APIs like `OracleCommand.ExecuteNonQuery()`, `OracleDataReader.Read()`, and `OracleDataAdapter.Fill`.

Declaration

```
// C#  
public bool EnableDBRoundTripTracing { get; set; }
```

Return Value

A bool.

Remarks

Default value is true.

This property enables instrumentation of database round-trips for the following public APIs:

- `OracleCommand.ExecuteNonQuery()`
- `OracleCommand.ExecuteReader` (All overloads)
- `OracleCommand.ExecuteScalar()`
- `OracleCommand.ExecuteScalarAsync` (All overloads)
- `OracleCommand.ExecuteStream()`
- `OracleCommand.ExecuteToStream(Stream outputStream)`
- `OracleCommand.ExecuteXmlReader()`
- `OracleCommand.ExecuteNonQueryAsync` (All overloads)
- `OracleCommand.ExecuteReaderAsync` (All overloads)
- `OracleCommand.ExecuteScalarAsync(CancellationToken cancellationToken)`
- `OracleCommand.ExecuteXmlReaderAsync` (All overloads)
- `OracleConnection.Open()`
- `OracleConnection.OpenAsync` (All overloads)
- `OracleConnection.Close()`
- `OracleDataReader.Read()`
- `OracleDataReader.ReadAsync` (All overloads)
- `OracleDataAdapter.Fill` (All overloads)

`Open()`, `OpenAsync()`, and `Close()` instrumentation requires the [EnableOpenCloseTracing](#) enum to be set to any valid value except `EnableOpenCloseTracing.None`.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataProviderInstrumentationOptions Class](#)
- [OracleDataProviderInstrumentationOptions Properties](#)

EnableOpenCloseTracing

This property indicates whether to record database soft and hard connection/disconnection instrumentation or not.

Declaration

```
// C#  
public EnableOpenCloseTracing EnableOpenCloseTracing { get; set; }
```

Return Value

EnableOpenCloseTracing enum values

Remarks

Default value is EnableOpenCloseTracing.None.

This option takes following EnableOpenCloseTracing enum values:

- EnableOpenCloseTracing.None: No open/close instrumentation
- EnableOpenCloseTracing.HardOpenCloseOnly: Hard open/close instrumentation only
- EnableOpenCloseTracing.AllOpenClose: All open/close (soft + hard) instrumentation

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataProviderInstrumentationOptions Class](#)
- [OracleDataProviderInstrumentationOptions Properties](#)

EnableSqlIdTracing

This property enables SQL identifier instrumentation of each executed SQL statement.

Declaration

```
// C#  
public bool EnableSqlIdTracing { get; set; }
```

Return Value

A bool.

Remarks

Default value is false.

This SQL identifier allows you to uniquely identify the SQL being executed through an identifier. The property's value is equivalent to the Oracle Database's statement SQL identifier (SQL_ID) so that the SQL execution can be matched between ODP.NET and server.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataProviderInstrumentationOptions Class](#)
- [OracleDataProviderInstrumentationOptions Properties](#)

InstrumentOracleDataReaderRead

This property indicates whether to instrument OracleDataReader Read method calls.

Declaration

```
// C#  
public bool InstrumentOracleDataReaderRead { get; set; }
```

Return Value

A bool.

Remarks

Default value is false.

Note

If EnableDBRoundTripTracing is set to false, then OracleDataReader Read() database round trips will not be instrumented.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataProviderInstrumentationOptions Class](#)
- [OracleDataProviderInstrumentationOptions Properties](#)

RecordException

This property indicates whether or not to record exception details in instrumentation.

Declaration

```
// C#  
public bool RecordException { get; set; }
```

Return Value

A bool.

Remarks

Default value is false.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataProviderInstrumentationOptions Class](#)
- [OracleDataProviderInstrumentationOptions Properties](#)

RequireApplicationRootSpanCreation

This property controls whether ODP.NET instruments only those operations that are associated with root spans created by the application or not.

Declaration

```
// C#  
public bool RequireApplicationRootSpanCreation { get; set; }
```

Return Value

A bool

Remarks

Default value is false.

When a new activity/span is created, it becomes child of current (in progress) activity/span automatically. But if there is no span in progress and a new span is created, then it will become Root/parentless/orphan span.

Similarly, if the user application creates a span before calling ODP.NET instrumented methods, the span created by the user application becomes root span and all span created by ODP.NET become children of it.

If user application did not create a root span and call ODP.NET instrumented methods, the multiple parentless/orphan spans will be created because of no parent linking. These spans are hard to visualize and analyze.

Parentless/Orphan span are also generated by server/database round trip calls happening from background threads. parentless/orphan spans do not add much value in most scenarios because of missing parent linking. Using this option user/client can control parentless/orphan span tracing as per their needs.

By default, instrumentation of parentless/orphan span is enabled.

This means application need not create any activity before calling ODP.NET instrumented methods. But if the user sets this option to true, then the user application must create a root activity, otherwise no ODP.NET spans will be generated.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataProviderInstrumentationOptions Class](#)
- [OracleDataProviderInstrumentationOptions Properties](#)

SetDbStatementForStoredProcedure

This property indicates whether or not to instrument stored procedure and function names when the command type is for a stored procedure (CommandType.StoredProcedure).

Declaration

```
// C#  
public bool SetDbStatementForStoredProcedure { get; set; }
```

Return Value

A bool.

Remarks

Default value is true.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataProviderInstrumentationOptions Class](#)
- [OracleDataProviderInstrumentationOptions Properties](#)

SetDbStatementForText

This property indicates whether or not to instrument SQL queries when command type is of Text type (CommandType.Text).

Declaration

```
// C#  
public bool SetDbStatementForText { get; set; }
```

Return Value

A bool.

Remarks

Default value is false.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataProviderInstrumentationOptions Class](#)
- [OracleDataProviderInstrumentationOptions Properties](#)

AddOracleDataProviderInstrumentation

Client applications call this extension method to enable OpenTelemetry instrumentation in ODP.NET. Certain tags and attributes are enabled/disabled based on the options provided.

Declaration

```
// C#
public static TracerProviderBuilder AddOracleDataProviderInstrumentation
(this TracerProviderBuilder builder, Action<OracleDataProviderInstrumentationOptions>
configureInstrumentationOptions = null)
```

Remarks

If no options are passed default options will be used.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OracleDataProviderInstrumentationOptions Class](#)
- [OracleDataProviderInstrumentationOptions Properties](#)

OraclePoolNameCollection Class

This class represents the pool names collection corresponding to each unique connection string.

Namespace: `Oracle.ManagedDataAccess.Client`

Assembly: `Oracle.ManagedDataAccess`

Declaration

```
// C#
public sealed class OraclePoolNameCollection
```

Inheritance

Object -> `OraclePoolNameCollection`

Thread Safety

This class is a thread-safe class.

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

Remarks

Connection strings combined with application names and process identifiers can be very long and hard to distinguish with many that look similar when using .NET Metrics or Windows Performance Counters. This collection class allows applications to set an arbitrary string to easily identify which pool is being monitored.

Constructor

Internal OraclePoolNameCollection() – This class is instantiated internally.

① See Also

- [Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)
- [OraclePoolNameCollection Methods](#)

OraclePoolNameCollection Methods

OraclePoolNameCollection methods are listed in OraclePoolNameCollection Methods.

Table 18-2 OraclePoolNameCollection Methods

Method	Description
Add(string, string)	Add an element with pool name as the key and connection string as the value
Remove(string)	Removes the element with the specified key from the collection

① Note

- [Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)
- [OraclePoolNameCollection Class](#)

Add(string, string)

This method adds an element with pool name as the key and connection string as the value.

Definition

```
C#  
public void Add(string poolName, string passwordLessConnectionString)
```

Parameters

- poolName (string) – name to identify pool with
- passwordLessConnectionString (string) – connection string without password field

Exceptions

NullReferenceException – If either or both poolName and passwordLessConnectionString are null.

See Also

- [Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)
- [OraclePoolNameCollection Class](#)
- [OraclePoolNameCollection Methods](#)

Remove(string)

This method removes the element with the specified key from the collection.

Definition

```
C#  
public void Remove(string poolName)
```

Parameters

- poolName (string) – name of the pool to remove

Exceptions

NullReferenceException – If poolName is null.

See Also

- [Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)
- [OraclePoolNameCollection Class](#)
- [OraclePoolNameCollection Methods](#)

Oracle Data Provider for .NET Oracle Identity and Access Management Classes

This chapter describes Oracle Identity and Access Management (IAM) single sign-on classes and objects. They are available through the `Oracle.ManagedDataAccess.Oci` NuGet package, which can be used with ODP.NET Core and managed ODP.NET.

- [OciTokenAuthentication Class](#)

① **See Also**

- [UseOciTokenAuthentication](#)
- [OciConfigurationFile](#)
- [OciProfile](#)

OciTokenAuthentication Class

This class accepts configuration parameters for OCI IAM SSO integration with ODP.NET. The class is part of the `Oracle.ManagedDataAccess.Oci` NuGet package. Applications provide the OCI IAM values in the class properties to connect. Once ODP.NET initiates opening the connection, the class instance is marked as read-only to prevent further modification during the pool's lifetime.

If customers use the same instance for another connection open, this new connection will be opened in the same connection pool as the earlier opened connection.

All the `OciTokenAuthentication` class properties should not be set to null nor empty values. Those null/empty values will be ignored, though no error is thrown.

Class Inheritance

`OracleInternal.Common.AbstractTokenAuthentication`

Declaration

```
// C#
public sealed class OciTokenAuthentication : AbstractTokenAuthentication
```

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client

Provider	ODP.NET, Managed Driver	ODP.NET Core
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

① See Also

- [Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)
- [OciTokenAuthentication Members](#)
- [OciTokenAuthentication Constructor\(\)](#)
- [OciTokenAuthentication Properties](#)

OciTokenAuthentication Members

OciTokenAuthentication members are listed in the following tables.

OciTokenAuthentication Constructors

OciTokenAuthentication constructors are listed in [Table 19-1](#).

Table 19-1 OciTokenAuthentication Constructors

Constructor	Description
OciTokenAuthentication Constructor()	Instantiates an empty object of the OciTokenAuthentication class

OciTokenAuthentication Properties

OciTokenAuthentication properties are listed in [Table 19-2](#).

Table 19-2 OciTokenAuthentication Properties

Property	Description
Compartment	Sets the database compartment's Oracle Cloud ID (OCID).
ConfigurationFile	Sets the configuration file location where the OCI token authentication values can be found.
Database	Sets the database Oracle Cloud ID (OCID).
Profile	Sets the OCI profile to use from the token authentication configuration file.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [AzureTokenAuthentication Class](#)

OciTokenAuthentication Constructor()

This constructor instantiates an empty object of the OciTokenAuthentication class.

Declaration

```
// C#
public OciTokenAuthentication();
```

Remarks

This constructor without parameters is invoked in the processing of object initializers. Parameters should be set individually as per the selected OCI IAM SSO authentication flow.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OciTokenAuthentication Class](#)
- [OciTokenAuthentication Members](#)

OciTokenAuthentication Properties

OciTokenAuthentication properties are listed in [Table 19-3](#).

Table 19-3 OciTokenAuthentication Properties

Property	Description
Compartment	Sets the database compartment's Oracle Cloud ID (OCID).
ConfigurationFile	Sets the configuration file location where the OCI token authentication values can be found.
Database	Sets the database Oracle Cloud ID (OCID).
Profile	Sets the OCI profile to use from the token authentication configuration file.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OciTokenAuthentication Class](#)
- [OciTokenAuthentication Members](#)

Compartment

This property sets the database compartment's Oracle Cloud ID (OCID).

Declaration

```
// C#  
public string Compartment { set; }
```

Exceptions

An `InvalidOperationException` will be raised if the application tries to modify an `ociTokenAuthenticationConfiguration` object by setting this property once a connection has opened using that configuration object.

Remarks

This parameter is applicable to the OCI API key, OCI interactive, OCI instance principal, and OCI delegation token authentication flows only and is optional. This property becomes mandatory if the `OciDatabase` property is set.

There is no default value for this parameter. If it is not set, ODP.NET requests access to all the databases within the OCI IAM user's tenancy.

Sample Code

```
OracleConnection conn = new OracleConnection(constr);  
conn.TokenAuthentication = OracleTokenAuth.OciInstancePrincipal;  
OciTokenAuthentication tokenconfig = new OciTokenAuthentication{  
    Compartment = "<COMPARTMENT OCID>",  
    Database = "<DATABASE OCID>"  
}  
conn.UseOciTokenAuthentication(tokenConfig);  
conn.Open();  
conn.Close();
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OciTokenAuthentication Class](#)
- [OciTokenAuthentication Members](#)

ConfigurationFile

This property sets the configuration file location where the OCI token authentication values can be found.

Declaration

```
// C#  
public string ConfigurationFile { set; }
```

Exceptions

An `InvalidOperationException` will be raised if the application tries to modify an `ociTokenAuthenticationConfiguration` object by setting this property once a connection has opened using that configuration object.

Remarks

This parameter is applicable to the OCI API key and OCI interactive authentication flows only and is optional.

Whenever this parameter is provided, the entire directory path including the configuration file name should be specified. ODP.NET does not use the standard file name if it is missing. Example of setting configuration file is C:\Users\admin\.oci\config. Skipping the file name config from the path like C:\Users\admin\.oci\, will throw an error.

If this parameter is not set, then ODP.NET uses the OCI configuration file in the default location: C:/<user-profile>/.oci/config. If the parameter is not set and the default configuration file is not found, then the user is prompted on the console to provide a region id for the OCI interactive authentication flow. A list of possible region ids will be displayed to the user. In the case of the OCI API key flow, the user will see an error.

On Windows, the Oracle HOME and USERPROFILE environment variables are used for the default directory location if they are set. Otherwise, Oracle HOMEPATH and HOMEDRIVE environment variables are used.

Sample Code

```
OracleConnection conn = new OracleConnection(constr);
conn.TokenAuthentication = OracleTokenAuth.OciApiKey;
OciTokenAuthentication tokenconfig = new OciTokenAuthentication{
    ConfigurationFile = "<CONFIGURATION FILE>",
    Profile = "<PROFILE>",
    Compartment = "<COMPARTMENT OCID>",
    Database = "<DATABASE OCID>"
}
conn.UseOciTokenAuthentication(tokenConfig);
conn.Open();
conn.Close();
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OciTokenAuthentication Class](#)
- [OciTokenAuthentication Members](#)

Database

This property sets the database Oracle Cloud ID (OCID).

Declaration

```
// C#
public string Database { get; set; }
```

Exceptions

An `InvalidOperationException` will be raised if the application tries to modify an `ociTokenAuthenticationConfiguration` object by setting this property once a connection has opened using that configuration object.

Remarks

This parameter is applicable to the OCI API key, OCI interactive, OCI instance principal, and OCI delegation token authentication flows only and is optional.

There is no default value for this parameter. If it is not set, ODP.NET requests access to all the compartment's databases specified by the `OciCompartment` property. If `OciCompartment` property is also not set, then ODP.NET requests access to all the databases within the OCI IAM user's tenancy.

Sample Code

```
OracleConnection conn = new OracleConnection(constr);
conn.TokenAuthentication = OracleTokenAuth.OciDelegationToken;
OciTokenAuthentication tokenconfig = new OciTokenAuthentication{
    Compartment = "<COMPARTMENT OCID>",
    Database = "<DATABASE OCID>"
}
conn.UseOciTokenAuthentication(tokenConfig);
conn.Open();
conn.Close();
```

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OciTokenAuthentication Class](#)
- [OciTokenAuthentication Members](#)

Profile

This property sets the OCI profile to use from the token authentication configuration file.

Declaration

```
// C#
public string Profile { set; }
```

Exceptions

An `InvalidOperationException` will be raised if the application tries to modify an `ociTokenAuthenticationConfiguration` object by setting this property once a connection has opened using that configuration object.

Remarks

This parameter is applicable to the OCI API key and OCI interactive authentication flows only and is optional.

ODP.NET uses the DEFAULT profile if no profile value is set.

Sample Code

```
OracleConnection conn = new OracleConnection(constr);
conn.TokenAuthentication = OracleTokenAuth.OciInteractive;
```

```
OciTokenAuthentication tokenconfig = new OciTokenAuthentication{
    ConfigurationFile = "<CONFIGURATION FILE>",
    Profile = "<PROFILE>",
    Compartment = "<COMPARTMENT OCID>",
    Database = "<DATABASE OCID>"
}
conn.UseOciTokenAuthentication(tokenConfig);
conn.Open();
conn.Close();
```

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [OciTokenAuthentication Class](#)
- [OciTokenAuthentication Members](#)

Oracle Data Provider for .NET Microsoft Entra ID Classes

This chapter describes the Microsoft Entra ID single sign-on classes and objects. They are available through the `Oracle.ManagedDataAccess.Azure` NuGet package. Microsoft Entra ID was formerly named Azure Active Directory.

- [AzureTokenAuthentication Class](#)

AzureTokenAuthentication Class

This class accepts configuration parameters for Microsoft Entra ID SSO integration with ODP.NET. The class is part of the `Oracle.ManagedDataAccess.Azure` NuGet package. Applications provide the Microsoft Entra ID values in the class properties to connect. Once ODP.NET initiates opening the connection, the class instance is marked as read-only to prevent further modification during the pool's lifetime.

If customers use same instance for another connection open, this new connection will be opened in the same connection pool as the earlier opened connection.

Class Inheritance

System.Object
System.Object.AbstractTokenAuthentication
Oracle.ManagedDataAccess.Azure.AzureTokenAuthentication

Declaration

```
// C#  
public class AzureTokenAuthentication : AbstractTokenAuthentication
```

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

① See Also

- [Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces](#)
- [AzureTokenAuthentication Members](#)

AzureTokenAuthentication Members

AzureTokenAuthentication members are listed in the following tables.

AzureTokenAuthentication Constructors

AzureTokenAuthentication constructors are listed in [Table 20-1](#).

Table 20-1 AzureTokenAuthentication Constructors

Constructor	Description
AzureTokenAuthentication()	Instantiates an empty object of the AzureTokenAuthentication class
AzureTokenAuthentication(string)	Instantiates an empty object of the AzureTokenAuthentication class with a database application id URI

AzureTokenAuthentication Properties

AzureTokenAuthentication properties are listed in [Table 20-2](#).

Table 20-2 AzureTokenAuthentication Properties

Property	Description
DatabaseApplicationIdUri	Sets the database application URI identifier, which is the protected resource identifier on Microsoft Entra ID for which client application requests the access token
ClientId	Sets the client identifier, which is Microsoft Entra ID registered application identifier
TenantId	Sets the Azure tenant identifier under which applications (client app or database) are registered
RedirectUri	Sets the reply URL where the authorization server sends the user once the app has been successfully authorized and granted an authorization code or access token
ClientSecret	Sets the Azure resource credential
ClientCertificate	This property sets the Azure resource digital certificate.
ClientCertificatePassword	Sets the Azure client certificate password
CustomDeviceCodeCallback	Sets a delegate/callback to the application providing them the Azure SDK DeviceCodeInfo object.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [AzureTokenAuthentication Class](#)

AzureTokenAuthentication Constructors

AzureTokenAuthentication constructors instantiate new instances of AzureTokenAuthentication class.

Overload List:

- [AzureTokenAuthentication\(\)](#)
This constructor instantiates an empty object of the AzureTokenAuthentication class.
- [AzureTokenAuthentication\(string\)](#)
This constructor instantiates an empty object of the AzureTokenAuthentication class with a database application id URI.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [AzureTokenAuthentication Class](#)
- [AzureTokenAuthentication Members](#)

AzureTokenAuthentication()

This constructor instantiates an empty object of the AzureTokenAuthentication class.

Declaration

```
// C#  
public AzureTokenAuthentication();
```

Remarks

This constructor without parameters is invoked in the processing of object initializers. Parameters should be set individually as per the selected Microsoft Entra ID authentication flow.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [AzureTokenAuthentication Class](#)
- [AzureTokenAuthentication Members](#)

AzureTokenAuthentication(string)

This constructor instantiates an empty object of the AzureTokenAuthentication class with a database application id URI.

Declaration

```
// C#  
public AzureTokenAuthentication(string databaseApplicationIdUri);
```

Parameters

databaseApplicationIdUri – The database application id URI configured in the database app registration on Microsoft Entra ID.

Exceptions

ArgumentException if databaseApplicationIdUri is null, empty, or has only whitespaces.

Remarks

The database application id URI is a mandatory parameter for all Microsoft Entra ID authentication flows. Other parameters should be set individually as per the selected authentication flow.

See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [AzureTokenAuthentication Class](#)
- [AzureTokenAuthentication Members](#)

AzureTokenAuthentication Properties

AzureTokenAuthentication properties are listed in [Table 20-3](#).

Table 20-3 AzureTokenAuthentication Properties

Property	Description
DatabaseApplicationIdUri	Sets the database application URI identifier, which is the protected resource identifier on Microsoft Entra ID for which client application requests the access token
ClientId	Sets the client identifier, which is Microsoft Entra ID registered application identifier
TenantId	Sets the Azure tenant identifier under which applications (client app or database) are registered
RedirectUri	Sets the reply URL where the authorization server sends the user once the app has been successfully authorized and granted an authorization code or access token
ClientSecret	Sets the Azure resource credential
ClientCertificate	This property sets the Azure resource digital certificate.

Table 20-3 (Cont.) AzureTokenAuthentication Properties

Property	Description
ClientCertificatePassword	Sets the Azure client certificate password
CustomDeviceCodeCallback	Sets a delegate/callback to the application providing them the Azure SDK DeviceCodeInfo object.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [AzureTokenAuthentication Class](#)
- [AzureTokenAuthentication Members](#)

DatabaseApplicationIdUri

This property sets the database application URI identifier, which is the protected resource identifier on Microsoft Entra ID for which client application requests the access token.

Declaration

```
// C#  
public string DatabaseApplicationIdUri { set; }
```

Exceptions

InvalidOperationException – if application tries to set this property once a connection is already opened using its encapsulating object.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [AzureTokenAuthentication Class](#)
- [AzureTokenAuthentication Members](#)

ClientId

This property sets the client identifier, which is Microsoft Entra ID registered application identifier.

Declaration

```
// C#  
public string ClientId { set; }
```

Exceptions

InvalidOperationException – if application tries to set this property once a connection is already opened using its encapsulating object.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [AzureTokenAuthentication Class](#)
- [AzureTokenAuthentication Members](#)

TenantId

This property sets the Azure tenant identifier under which applications (client app or database) are registered.

Declaration

```
// C#  
public string TenantId { set; }
```

Exceptions

InvalidOperationException – if application tries to set this property once a connection is already opened using its encapsulating object.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [AzureTokenAuthentication Class](#)
- [AzureTokenAuthentication Members](#)

RedirectUri

This property sets the reply URL where the authorization server sends the user once the app has been successfully authorized and granted an authorization code or access token.

Declaration

```
// C#  
public string RedirectUri { set; }
```

Exceptions

InvalidOperationException – if application tries to set this property once a connection is already opened using its encapsulating object.

Remarks

This property is used with Azure interactive authentication.

This property has "http://localhost" as the default value if not provided by the user.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [AzureTokenAuthentication Class](#)
- [AzureTokenAuthentication Members](#)

ClientSecret

This property sets the Azure resource credential.

Declaration

```
// C#  
public SecureString ClientSecret { set; }
```

Exceptions

InvalidOperationException – if application tries to set this property once a connection is already opened using its encapsulating object.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [AzureTokenAuthentication Class](#)
- [AzureTokenAuthentication Members](#)

ClientCertificate

This property sets the Azure resource digital certificate.

Declaration

```
// C#  
public string ClientCertificate { set; }
```

Exceptions

InvalidOperationException – if application tries to set this property once a connection is already opened using its encapsulating object.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [AzureTokenAuthentication Class](#)
- [AzureTokenAuthentication Members](#)

ClientCertificatePassword

This property sets the Azure client certificate password.

Declaration

```
// C#  
public SecureString ClientCertificatePassword { set; }
```

Exceptions

InvalidOperationException – if application tries to set this property once a connection is already opened using its encapsulating object.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [AzureTokenAuthentication Class](#)
- [AzureTokenAuthentication Members](#)

CustomDeviceCodeCallback

This property sets a delegate/callback to the application providing them the Azure SDK DeviceCodeInfo object.

Declaration

```
// C#  
public Func<DeviceCodeInfo, Task> CustomDeviceCodeCallback { set; }
```

Exceptions

InvalidOperationException – if application tries to set this property once a connection is already opened using its encapsulating object.

Remarks

This property is used with Azure device code authentication flow.

This property facilitates ODP.NET applications to receive display device code, URL, and other device code parameters. The DeviceCodeInfo object encapsulates these parameters for Azure use. Applications may choose to display these details if needed in the application user interface.

If this property is not set when using Azure device code authentication, then ODP.NET displays the device code and URL on the application console.

① See Also

- ["Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"](#)
- [AzureTokenAuthentication Class](#)
- [AzureTokenAuthentication Members](#)

Oracle Database Vector Store Connector Classes

Semantic Kernel is a lightweight, open-source development kit that lets you easily build AI agents and integrate the latest AI models into your C#, Python, or Java codebase. It serves as an efficient middleware that enables rapid delivery of enterprise-grade solutions. Semantic Kernel supports vector databases for storing information to enable large language models (LLMs) to generate more relevant and coherent responses.

.NET provides vector store abstractions as a common API set for adding and retrieving data from vector stores. Semantic Kernel uses these APIs for vector manipulation, but they can also be used by generic .NET vectors or any other framework that incorporates vectors. The Oracle Database Vector Store Connector for Semantic Kernel implements the .NET vector API set for adding data, retrieving data, and performing vector search in the Oracle database. Thus, it can support .NET vector and Semantic Kernel applications. This chapter describes the connector classes.

ODP.NET Core or managed ODP.NET provides connector data access. The `Oracle.VectorData` NuGet package is required to use the connector.

- [OracleCollection Class](#)
- [OracleCollectionOptions Class](#)
- [OracleDynamicCollection Class](#)
- [OracleServiceCollectionExtensions Class](#)
- [OracleVectorStore Class](#)
- [OracleVectorStoreOptions Class](#)

See Also

- [Microsoft Semantic Kernel](#)
- [Semantic Kernel Vector Stores](#)
- [Using the Oracle Database Vector Store connector \(Preview\)](#)

OracleCollection Class

An `OracleCollection` object implements the `Microsoft.Extensions.VectorData.VectorStoreCollection` interface. It is used to search and manage records, and to create or delete the collection.

In an Oracle database, a named collection is mapped to an Oracle database table, and the table definition is also defined based on the provided key type and the data model. A collection record is mapped to a database row of an Oracle database table.

Class Inheritance

System.Object
 Microsoft.Extensions.VectorData.VectorStoreCollection
 Oracle.VectorData.OracleCollection

Declaration

```
// C#  
public class OracleCollection :  
Microsoft.Extensions.VectorData.VectorStoreCollection
```

Thread Safety

All public methods are thread-safe, unless stated otherwise.

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

 See Also

- [OracleCollection Constructors](#)
- [OracleCollection Properties](#)
- [OracleCollection Public Methods](#)

OracleCollection Constructors

The OracleCollection constructors create new instances of the OracleCollection class with the provided data source or connection string and the collection name.

Overload List:

- [OracleCollection<TKey, TRecord>\(OracleDataSource, string, OracleCollectionOptions?\)](#)
This constructor instantiates a new instance of OracleCollection class using the provided data source and collection name. It is available to use only in a .NET Core environment.
- [OracleCollection<TKey, TRecord>\(string, string, OracleCollectionOptions?\)](#)
This constructor creates a new instance of the OracleCollection class using the provided connection string and collection name.

① See Also

- [OracleCollection Class](#)
- [OracleCollection Properties](#)
- [OracleCollection Public Methods](#)

OracleCollection<TKey, TRecord>(OracleDataSource, string, OracleCollectionOptions?)

Instantiates a new instance of OracleCollection class using the provided data source and collection name.

Declaration

```
// C#
OracleCollection<TKey, TRecord>(OracleDataSource dataSource, string
collectionName, OracleCollectionOptions? options = default)
```

Parameters

- **dataSource**
The data source which can be used to obtain open database connections.
- **collectionName**
Name of the collection
- **options**
Optional vector collection configuration options.

Exceptions

- **ArgumentNullException:** The provided dataSource or name parameter is null.
- **ArgumentException:**
 - The provided collectionName parameter is null, an empty string or contains white spaces only.
 - The provided collectionName is not compliant with Oracle name standard.
 - The definition provided in OracleCollectionOptions.Definition cannot be mapped to an Oracle type based on the default/custom type mapping.

① See Also

- [OracleDataSource Class](#)
- [OracleCollectionOptions Class](#)

OracleCollection<TKey, TRecord>(string, string, OracleCollectionOptions?)

Instantiates a new instance of OracleCollection class using the provided connection string and collection name.

Declaration

```
// C#  
OracleCollection<TKey, TRecord>(string connectionString, string  
collectionName, OracleCollectionOptions? options = default)
```

Parameters

- **string**
The connection string which can be used to obtain database connections.
- **collectionName**
Name of the collection
- **options**
Optional vector collection configuration options.

Exceptions

- **ArgumentNullException:** The provided dataSource or name parameter is null.
- **ArgumentException:**
 - The provided collectionName parameter is null, an empty string or contains white spaces only.
 - The provided collectionName is not compliant with Oracle name standard.
 - The definition provided in OracleCollectionOptions.Definition cannot be mapped to an Oracle type based on the default/custom type mapping.

① See Also

- [OracleConnection.ConnectionString](#)
- [OracleCollectionOptions Class](#)

OracleCollection Properties

OracleCollection properties are listed in [Table 21-1](#).

Table 21-1 OracleCollection Properties

Property	Description
Name	This property specifies the name of this collection/table in the Oracle database.

① See Also

- [OracleCollection Class](#)
- [OracleCollection Constructors](#)
- [OracleCollection Public Methods](#)

Name

This property specifies the name of this collection/table in the Oracle database.

Declaration

```
// C#
public string Name { get; }
```

Property Value

The string specifying the name of the collection. The collection name is the name given by the application when the OracleCollection instance is created.

Implements

```
Microsoft.Extensions.VectorData.VectorStoreCollection
```

OracleCollection Public Methods

OracleCollection public methods are listed in [Table 21-2](#).

Table 21-2 OracleCollection Public Methods

Public Method	Description
CollectionExistsAsync	Checks whether the collection exists in the vector store.
DeleteAsync	Deletes a single record or a batch of records from the vector store. It does not guarantee that the collection exists.
Dispose	Releases any resources or memory held by the object.
EnsureCollectionExistsAsync	Ensures the existence of the collection. Creates the collection in the vector store if it does not exist.
EnsureCollectionDeletedAsync	Deletes the collection from the vector store if it exists.
GetAsync	Retrieves a single record or a batch of records from the vector store. It does not guarantee that the collection exists and returns null if the record is not found.
GetService	Retrieves an object of a specified service type from the collection or vector store.
SearchAsync	Searches the vector store for records that are similar to input data.

Table 21-2 (Cont.) OracleCollection Public Methods

Public Method	Description
UpsertAsync	Updates or inserts a record or a batch of records into the vector store. It does not guarantee the existence of the collection. If the record already exists, it is updated; if it does not exist, it is created.

 See Also

- [OracleCollection Class](#)
- [OracleCollection Constructors](#)
- [OracleCollection Properties](#)

CollectionExistsAsync

This method checks if a collection exists in the vector store.

Declaration

```
// C#
public async Task<bool> CollectionExistsAsync(CancellationTok
cancellationToken = default)
```

Parameters

- `cancellationToken`
The cancellation token.

Return Value

A task representing the asynchronous operation. The task result contains a boolean indicating whether the collection exists in the database. True if exists, else false.

Implements

Microsoft.Extensions.VectorData.VectorStore

DeleteAsync

DeleteAsync methods delete a single record or a batch of records from the vector store. It does not guarantee that the collection exists.

Overload List

- [DeleteAsync\(TKey key, CancellationToken\)](#)
This method deletes a record from the vector store. It does not guarantee that the collection exists.
- [DeleteAsync\(IEnumerable<TKey> keys, CancellationToken\)](#)
This method deletes a batch of records from the vector store. It does not guarantee that the collection exists.

DeleteAsync(TKey key, CancellationToken)

This method deletes a record from the vector store. It does not guarantee that the collection exists.

Declaration

```
// C#  
public async Task DeleteAsync(<TKey> key, CancellationToken cancellationToken  
= default);
```

Parameters

- **key**
The unique key associated with the record to delete.
- **cancellationToken**
The cancellation token.

Return Value

A task representing the asynchronous operation that completes when the record is deleted.

Implements

Microsoft.Extensions.VectorData.VectorStoreCollection

Exceptions

VectorStoreException: The command fails to execute for any reason other than the absence of a record.

DeleteAsync(IEnumerable<TKey> keys, CancellationToken)

This method deletes a batch of records from the vector store. It does not guarantee that the collection exists.

Declaration

```
// C#  
public async Task DeleteAsync(IEnumerable<Tkey> keys, CancellationToken  
cancellationToken = default)
```

Parameters

- **keys**
The unique keys associated with the records to delete.
- **cancellationToken**
The cancellation token.

Return Value

A task representing the asynchronous operation when the records are deleted.

Implements

Microsoft.Extensions.VectorData.VectorStoreCollection

Exceptions

VectorStoreException: The command fails to execute for any reason other than the absence of a record.

Remarks

If a record is not found, it is ignored, and the batch operation succeeds. However, if a record cannot be deleted for any reason, the operation throws an error. Since some records may have already been deleted while others have not, the entire operation needs to be retried.

Dispose

This method releases any resources or memory allocated by the object.

Declaration

```
// C#  
public void Dispose()
```

Implements

System.IDisposable

Thread Safety

This method is not thread-safe.

EnsureCollectionDeletedAsync

This method deletes the collection from the vector store if it exists.

Declaration

```
// C#  
public async Task EnsureCollectionDeleteAsync(CancellationToken  
cancellation_token = default)
```

Parameters

- `cancellation_token`
The cancellation token.

Return Value

A task representing the asynchronous operation that completes when the collection has been deleted from the database.

Implements

Microsoft.Extensions.VectorData.VectorStoreCollection

EnsureCollectionExistsAsync

This method ensures the existence of the collection. It creates the collection in the vector store if it does not exist.

Declaration

```
// C#  
public async Task EnsureCollectionExistsAsync(CancellationTok  
cancellationTok = default)
```

Parameters

- `cancellationTok`
The cancellation token.

Return Value

A task representing the asynchronous operation that completes when the collection has been created in the database.

Implements

Microsoft.Extensions.VectorData.VectorStoreCollection

GetAsync

GetAsync methods retrieve a single record or a batch of records from the vector store. It does not guarantee that the collection exists and returns null if the record is not found.

Overload List

- [GetAsync\(TKey, RecordRetrieveOptions?, CancellationTok\)](#)
This method retrieves a single record from the vector store. It does not guarantee that the collection exists and returns null if the record is not found.
- [GetAsync\(IEnumerable<TKey>, RecordRetrieveOptions?, CancellationTok\)](#)
This method retrieves a batch of records from the vector store. It does not guarantee that the collection exists and returns null if the record is not found.
- [GetAsync\(Expression<Func<TRecord, bool>>, int, FilteredRecordRetrievalOptions<TRecord>?, CancellationTok\)](#)
This method retrieves from the vector store records matching a filter expression. It does not guarantee the existence of the collection.

GetAsync(TKey, RecordRetrieveOptions?, CancellationTok)

This method retrieves a single record from the vector store. It does not guarantee that the collection exists and returns null if the record is not found.

Declaration

```
// C#  
public async Task<TRecord> GetAsync(TKey key, RecordRetrieveOptions? options  
= null, CancellationTok cancellationTok = default);
```

Parameters

- **key**
The unique ID associated with the record to retrieve.
- **options**
Optional options for retrieving the records.
- **cancellationToken**
The cancellation token.

Return Value

Task<TRecord?> object representing the record if found, or null if not found.

Implements

Microsoft.Extensions.VectorData.VectorStoreCollection

Exceptions

VectorStoreException: The command fails to execute for any reason other than the absence of a record.

See Also

- [RecordRetrievalOptions](#)

GetAsync(IEnumerable<TKey>, RecordRetrieveOptions?, CancellationToken)

This method retrieves a batch of records from the vector store. It does not guarantee that the collection exists and returns null if the record is not found.

Declaration

```
// C#
public async IAsyncEnumerable<TRecord?> GetAsync(IEnumerable<TKey> keys,
RecordRetrieveOptions? options = null, CancellationToken cancellationToken =
default);
```

Parameters

- **key**
The unique IDs associated with the records to retrieve.
- **options**
Optional options for retrieving the records.
- **cancellationToken**
The cancellation token.

Return Value

IAsyncEnumerable<TRecord?> object representing the result set of found records.

Implements

Microsoft.Extensions.VectorData.VectorStoreCollection

Exceptions

VectorStoreException: The command fails to execute for any reason other than the absence of a record.

① See Also

- [RecordRetrievalOptions](#)

GetAsync(Expression<Func<TRecord, bool>>, int, FilteredRecordRetrievalOptions<TRecord>?, CancellationToken)

This method retrieves from the vector store records matching a filter expression. It does not guarantee the existence of the collection.

Declaration

```
// C#
public async IEnumerable<TRecord?> GetAsync( Expression<Func<TRecord,
bool>> filter, int top, FilteredRecordRetrievalOptions<TRecord>? options =
null, CancellationToken cancellationToken = default);
```

Parameter

- filter
Predicate for filtering the records.
- top
Maximum number of results to return.
- options
Optional options for retrieving the records.
- cancellationToken
The cancellation token.

Return Value

IAsyncEnumerable<TRecord?> object representing the records matching the provided predicate.

Implements

Microsoft.Extensions.VectorData.VectorStoreCollection

Exceptions

VectorStoreException: The command fails to execute for any reason other than the absence of a record.

Remarks

The filter cannot compare the following unsupported columns types:

- BFile
- Blob
- Clob
- Json
- NClob
- Vector
- Vector_Binary
- Vector_Float32
- Vector_Float64
- Vector_Int8
- XmlType

① See Also

- [FilteredRecordRetrievalOptions<TRecord> Class](#)

GetService

This method retrieves an object of a specified service type from the vector store.

Declaration

```
// C#  
public object? GetService(Type serviceType, object? serviceKey = null);
```

Parameters

- serviceType
The type of an object to retrieve. Supported ServiceType: VectorStoreCollectionMetaData, OracleCollection or OracleDataSource (.NET Core only).
- serviceKey
An optional key that can be used to identify the target service.

Return Value

An Object of a specified service type, otherwise null.

Implements

Microsoft.Extensions.VectorData.VectorStoreCollection

SearchAsync

This method searches the vector store for records that are similar to input data.

Declaration

```
// C#  
public async IEnumerable<VectorSearchResult<TRecord>>  
SearchAsync<TInput>(TInput searchValue, int top,  
VectorSearchOptions<TRecord>? options = null, CancellationToken  
cancellationToken = default) where TInput : notnull;
```

Parameters

- **TInput**
The type of the input value to perform the similarity search.
- **searchValue**
Input value for similarity search. For more information, see the Remarks section.
- **top**
The maximum number of results to return.
- **options**
The options that control search behavior.
- **cancellationToken**
The cancellation token.

Return Value

`IEnumerable<VectorSearchResult<TRecord>>` records found by the vector search, including their result scores.

Implements

`Microsoft.Extensions.VectorData.IVectorSearchable`

Exceptions

- **VectorStoreException**: The command fails to execute for any reason other than the absence of a record.
- **NotSupportedException**: The database is not a 26ai or later database that does not have vector data type support.

Remarks

The types supported for the `searchValue` parameter vary based on the provider being used and the embedding generation configured.

A string or [DataContent](#)(for images, sound...) if an appropriate [IEmbeddingGenerator](#) has been configured that accepts that type as input. For example, register an [IEmbeddingGenerator](#) that accepts string as input in your dependency injection container, and then pass in a string argument to this method; the argument will be automatically passed to the [IEmbeddingGenerator](#) to generate the embedding and perform the search.

Some databases support generating embeddings at the database side. In this case, you can pass in a string or [DataContent](#) without configuring an [IEmbeddingGenerator](#) with

[Microsoft.Extensions.VectorData](#). The provider will simply send your argument to the database as-is for embedding generation.

Arbitrary .NET types can also be passed in as long as an appropriate [IEmbeddingGenerator](#) has been configured; for example, you can create your own [IEmbeddingGenerator](#) that accepts your own custom types as input, and uses another [IEmbeddingGenerator](#) to generate embedding from multiple properties.

For .NET types beyond string and [DataContent](#), you must use the generic [VectorStoreVectorProperty](#) in your record definition.

To work with embeddings directly, pass in the .NET data types that the connector accepts for Vector data type.

If you are using [IEmbeddingGenerator](#) directly in your code, that type returns an [Embedding](#) (for example, `Embedding`), which can also be passed in directly, as long as the provider supports the specific embedding type. However, consider registering your [IEmbeddingGenerator](#) with the provider instead and pass in the input type (for example, string).

See Also

- [VectorSearchOptions<TRecord> Class](#)

UpsertAsync

`UpsertAsync` methods update or insert a record or a batch of records into the vector store. It does not guarantee the existence of the collection. If the record already exists, it is updated; if it does not exist, it is created.

Overload List

- [UpsertAsync\(TRecord, CancellationToken\)](#)
This method updates or inserts a record into the vector store. It does not guarantee the existence of the collection. If the record already exists, it is updated; if it does not exist, it is created.
- [UpsertAsync\(IEnumerable<TRecord>, CancellationToken\)](#)
This method updates or inserts a batch of records into the vector store. It does not guarantee the existence of the collection. If the record already exists, it is updated; if it does not exist, it is created.

UpsertAsync(TRecord, CancellationToken)

This method updates or inserts a record into the vector store. It does not guarantee the existence of the collection. If the record already exists, it is updated; if it does not exist, it is created.

Declaration

```
// C#
public async Task UpsertAsync(TRecord record, CancellationToken
cancellationTokn = default);
```

Parameters

- record
The record to update or insert.
- cancellationToken
The cancellation token.

Return Value

Task representing the asynchronous operation when the records are updated or inserted.

Implements

Microsoft.Extensions.VectorData.VectorStoreCollection

Exceptions

VectorStoreException: The command fails to execute for any reason other than the absence of a record.

UpsertAsync(IEnumerable<TRecord>, CancellationToken)

This method updates or inserts a batch of records into the vector store. It does not guarantee the existence of the collection. If the record already exists, it is updated; if it does not exist, it is created.

Declaration

```
// C#  
public async Task UpsertAsync(IEnumerable<TRecord> records, CancellationToken  
cancellationTokn = default);
```

Parameters

- records
The records to update or insert.
- cancellationToken
The cancellation token.

Return Value

Task representing the asynchronous operation when the records are updated or inserted.

Implements

Microsoft.Extensions.VectorData.VectorStoreCollection

Exceptions

VectorStoreException: The command fails to execute for any reason other than the absence of a record.

Remarks

All records are upserted when succeed or none when fails.

OracleCollectionOptions Class

The OracleCollectionOptions object encapsulates configuration options for creating an instance of the OracleCollection class. While creating the collection, these options control aspects like schema assignment, custom collection definitions, and embedding generator assignment.

Class Inheritance

```
System.Object
    Microsoft.Extensions.VectorData.VectorStoreCollectionOptions
        Oracle.VectorData.OracleCollectionOptions
```

Declaration

```
// C#
public class OracleCollectionOptions :
    Microsoft.Extensions.VectorData.VectorStoreCollectionOptions
```

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

 See Also

- [OracleCollectionOptions Constructor](#)
- [OracleCollectionOptions Properties](#)

OracleCollectionOptions Constructor

Instantiates a new instance of the OracleCollectionOptions class for configuring collection-level options.

Declaration

```
// C#
public OracleCollectionOptions()
```

① See Also

- [OracleCollectionOptions Class](#)
- [OracleCollectionOptions Properties](#)

OracleCollectionOptions Properties

OracleCollectionOptions properties are listed in [Table 21-3](#).

Table 21-3 OracleCollectionOptions Properties

Property	Description
EmbeddingGenerator	This property specifies the default embedding generator to use for generating vector embeddings when working with the collection.
Schema	This property specifies the database schema name under which collections are created by default.
Definition	This property specifies an optional record definition, describing the schema of the record type.

① See Also

- [OracleCollectionOptions Class](#)
- [OracleCollectionOptions Constructor](#)

EmbeddingGenerator

This property specifies the default embedding generator to use for generating vector embeddings when working with the collection.

Declaration

```
// C#
public IEmbeddingGenerator? EmbeddingGenerator { get; set; }
```

Property Value

The default embedding generator to use for generating vector embeddings when working with the vector store.

Implements

Inherited from `Microsoft.Extensions.VectorData.VectorStoreCollectionOptions`

Remarks

When embedding is specified, the collection uses the EmbeddingGenerator to generate embeddings. The default property value is null. When the embedding generator is provided, the collection automatically generates the embedding during both upsert and search operations, eliminating the need for the application to generate and provide the embedding.

Schema

This property specifies the database schema name under which collections are created.

Declaration

```
// C#  
public string? Schema { get; set; }
```

Property Value

The string specifying the name of the database schema.

Remarks

When Schema is specified, the collections under this vector store are created under the schema. The default schema value is null. When null, the collections are created under the connected user's schema.

Definition

This property specifies an optional record definition, describing the schema of the record type.

Declaration

```
// C#  
public VectorStoreCollectionDefinition? Definition { get; set; }
```

Property Value

An optional record Definition, describing the schema of the record type.

Implements

Inherited from `Microsoft.Extensions.VectorData.VectorStoreCollectionOptions`

Remarks

If not provided, schema is auto-inferred from the record model class. In this case, the record model properties needs to be annotated with the appropriate attributes to indicate their usage.

See Also

- [Defining your storage schema using a record definition](#)
- [VectorStoreCollectionDefinition Class](#)

OracleDynamicCollection Class

An `OracleDynamicCollection` object implements `Microsoft.Extensions.VectorData.VectorStoreCollection` interface. It is used to search and manage records, and to create or delete the collection itself.

The definition of the named collection is based on the application-provided key type and a generic data model, Dictionary<object, object?>. In an Oracle database, a named collection is mapped to an Oracle database table, and the table definition is defined based on the provided key type and the provided record definition.

An OracleDynamicCollection can be used without requiring an application to provide a data model. Records can be managed or processed using the generic Dictionary<string, object?> class.

Class Inheritance

```
System.Object
    Microsoft.Extensions.VectorData.VectorStoreCollection
        Oracle.VectorData.OracleCollection
            Oracle.VectorData.OracleDynamicCollection
```

Declaration

```
// C#
public class OracleDynamicCollection : OracleCollection <object,
Dictionary<string, object?>>
```

Thread Safety

All public methods are thread-safe, unless stated otherwise.

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

See Also

- [OracleDynamicCollection Constructors](#)
- [OracleDynamicCollection Properties](#)
- [OracleCollection Public Methods](#)
- [Using Vector Store abstractions without defining your own data model](#)

OracleDynamicCollection Constructors

The OracleDynamicCollection constructors create new instances of the OracleDynamicCollection class with the provided data source or connection string and the collection name.

Overload List:

- [OracleDynamicCollection\(OracleDataSource, string, OracleCollectionOptions\)](#)
This constructor instantiates a new instance of OracleDynamicCollection class using the provided data source and collection name
- [OracleDynamicCollection\(string, string, OracleCollectionOptions\)](#)
This constructor instantiates a new instance of OracleDynamicCollection class using the provided connection string and collection name.

See Also

- [OracleDynamicCollection Class](#)
- [OracleDynamicCollection Properties](#)
- [OracleDynamicCollection Public Methods](#)

OracleDynamicCollection(OracleDataSource, string, OracleCollectionOptions)

Instantiates a new instance of OracleDynamicCollection class using the provided data source and collection name.

Declaration

```
// C#
OracleDynamicCollection (OracleDataSource dataSource, string collectionName,
OracleCollectionOptions options)
```

Parameters

- `dataSource`
The data source which can be used to obtain open database connections.
- `collectionName`
Name of the collection
- `options`
Vector collection configuration options.

Exceptions

- `ArgumentNullException`: The provided `dataSource` or `name` parameter is null.
- `ArgumentException`:
 - The provided `collectionName` parameter is null, an empty string or contains white spaces only.
 - The provided `collectionName` is not compliant with Oracle name standard.
 - The definition provided in `OracleCollectionOptions.Definition` is null or cannot be mapped to an Oracle type based on the default/custom type mapping.

① See Also

- [OracleDataSource Class](#)
- [GetDynamicCollection\(string, VectorStoreCollectionDefinition\)](#)
- [OracleCollectionOptions Class](#)

OracleDynamicCollection(string, string, OracleCollectionOptions)

Instantiates a new instance of `OracleDynamicCollection` class using the provided connection string and collection name.

Declaration

```
// C#
OracleDynamicCollection(string connectionString, string collectionName,
OracleCollectionOptions options)
```

Parameters

- `string`
The connection string which can be used to obtain database connections.
- `collectionName`
Name of the collection
- `options`
Vector collection configuration options.

Exceptions

- `ArgumentNullException`: The provided `dataSource` or `name` parameter is null.
- `ArgumentException`:
 - The provided `collectionName` parameter is null, an empty string or contains white spaces only.
 - The provided `collectionName` is not compliant with Oracle name standard.
 - The definition provided in `OracleCollectionOptions.Definition` is null or cannot be mapped to an Oracle type based on the default/custom type mapping.

① See Also

- [OracleDataSource Class](#)
- [GetDynamicCollection\(string, VectorStoreCollectionDefinition\)](#)
- [OracleCollectionOptions Class](#)

OracleDynamicCollection Properties

OracleDynamicCollection properties are listed in [Table 21-4](#).

Table 21-4 OracleDynamicCollection Properties

Property	Description
Name	This property specifies the name of this collection/table in the Oracle database.

See Also

- [OracleDynamicCollection Class](#)
- [OracleDynamicCollection Constructors](#)
- [OracleDynamicCollection Public Methods](#)

Name

This property specifies the name of this collection/table in the Oracle database.

Declaration

```
// C#  
public string Name { get; }
```

Property Value

The string specifying the name of the collection. The collection name is the name given by the application when the OracleDynamicCollection instance is created.

Implements

Microsoft.Extensions.VectorData.VectorStoreCollection

OracleDynamicCollection Public Methods

OracleDynamicCollection public methods are listed in [Table 21-5](#).

Table 21-5 OracleDynamicCollection Public Methods

Public Method	Description
CollectionExistsAsync	Checks whether the collection exists in the vector store.
DeleteAsync	Deletes a single record or a batch of records from the vector store. It does not guarantee that the collection exists.
Dispose	Releases any resources or memory held by the object.

Table 21-5 (Cont.) OracleDynamicCollection Public Methods

Public Method	Description
EnsureCollectionExistsAsync	Ensures the existence of the collection. Creates the collection in the vector store if it does not exist.
EnsureCollectionDeletedAsync	Deletes the collection from the vector store if it exists.
#unique_2991	Retrieves a single record or a batch of records from the vector store. It does not guarantee that the collection exists and returns null if the record is not found.
GetService	Retrieves an object of a specified service type from the collection or vector store.
SearchAsync	Searches the vector store for records that are similar to input data.
UpsertAsync	Updates or inserts a record or a batch of records into the vector store. It does not guarantee the existence of the collection. If the record already exists, it is updated; if it does not exist, it is created.

See Also

- [OracleDynamicCollection Class](#)
- [OracleDynamicCollection Constructors](#)
- [OracleDynamicCollection Properties](#)

GetAsync

GetAsync methods retrieve a single record or a batch of records from the vector store. It does not guarantee that the collection exists and returns null if the record is not found.

Overload List

- [GetAsync\(Object, RecordRetrieveOptions?, CancellationToken\)](#)
This method retrieves a single record from the vector store. It does not guarantee that the collection exists and returns null if the record is not found.
- [GetAsync\(IEnumerable<Object>, RecordRetrieveOptions?, CancellationToken\)](#)
This method retrieve a batch of records from the vector store. It does not guarantee that the collection exists and returns null if the record is not found.
- [GetAsync\(Expression<Func<Dictionary<string, object?>, bool>>, int, <Dictionary<string, object?>>?, CancellationToken\)](#)
This method retrieves from the vector store records matching a filter expression. It does not guarantee the existence of the collection.

CollectionExistsAsync

This method checks if a collection exists in the vector store.

Declaration

```
// C#  
public async Task<bool> CollectionExistsAsync(CancellationTok  
cancellationToken = default)
```

Parameters

- `cancellationToken`
The cancellation token.

Return Value

A task representing the asynchronous operation. The task result contains a boolean indicating whether the collection exists in the database. True if exists, else false.

Implements

Microsoft.Extensions.VectorData.VectorStore

DeleteAsync

DeleteAsync methods delete a single record or a batch of records from the vector store. It does not guarantee that the collection exists.

Overload List

- [DeleteAsync\(Object key, CancellationToken\)](#)
This method deletes a record from the vector store. It does not guarantee that the collection exists.
- [DeleteAsync\(IEnumerable<Object> keys, CancellationToken\)](#)
This method deletes a batch of records from the vector store. It does not guarantee that the collection exists.

DeleteAsync(Object key, CancellationToken)

This method deletes a record from the vector store. It does not guarantee that the collection exists.

Declaration

```
// C#  
public async Task DeleteAsync(Object key, CancellationToken cancellatio  
= default);
```

Parameters

- `key`
The unique key associated with the record to delete.
- `cancellationToken`
The cancellation token.

Return Value

A task representing the asynchronous operation that completes when the record is deleted.

Implements

Microsoft.Extensions.VectorData.VectorStoreCollection

Exceptions

VectorStoreException: The command fails to execute for any reason other than the absence of a record.

Remarks

Unlike OracleCollection, OracleDynamicCollection uses Object as TKey, and Dictionary<string, object?> as TRecord. Please make sure to use the correct parameter types.

DeleteAsync(IEnumerable<Object> keys, CancellationToken)

This method deletes a batch of records from the vector store. It does not guarantee that the collection exists.

Declaration

```
// C#  
public async Task DeleteAsync(IEnumerable<Object> keys, CancellationToken  
cancellationTok = default);
```

Parameters

- **keys**
The unique keys associated with the records to delete.
- **cancellationTok**
The cancellation token.

Return Value

A task representing the asynchronous operation when the records are deleted.

Implements

Microsoft.Extensions.VectorData.VectorStoreCollection

Exceptions

VectorStoreException: The command fails to execute for any reason other than the absence of a record.

Remarks

If a record is not found, it is ignored, and the batch operation succeeds. However, if a record cannot be deleted for any reason, the operation throws an error. Since some records may have already been deleted while others have not, the entire operation needs to be retried.

Unlike `OracleCollection`, `OracleDynamicCollection` uses `Object` as `TKey`, and `Dictionary<string, object?>` as `TRecord`. Please make sure to use the correct parameter types: `IEnumerable<Object>` for keys.

Dispose

This method releases any resources or memory allocated by the object.

Declaration

```
// C#  
public void Dispose()
```

Implements

`System.IDisposable`

Thread Safety

This method is not thread-safe.

EnsureCollectionDeletedAsync

This method deletes the collection from the vector store if it exists.

Declaration

```
// C#  
public async Task EnsureCollectionDeleteAsync(CancellationTokentoken  
cancellationTokentoken = default)
```

Parameters

- `cancellationTokentoken`
The cancellation token.

Return Value

A task representing the asynchronous operation that completes when the collection has been deleted from the database.

Implements

`Microsoft.Extensions.VectorData.VectorStoreCollection`

EnsureCollectionExistsAsync

This method ensures the existence of the collection. It creates the collection in the vector store if it does not exist.

Declaration

```
// C#  
public async Task EnsureCollectionExistsAsync(CancellationTokentoken  
cancellationTokentoken = default)
```

Parameters

- `cancellationToken`
The cancellation token.

Return Value

A task representing the asynchronous operation that completes when the collection has been created in the database.

Implements

`Microsoft.Extensions.VectorData.VectorStoreCollection`

GetAsync(Object, RecordRetrieveOptions?, CancellationToken)

This method retrieves a single record from the vector store. It does not guarantee that the collection exists and returns null if the record is not found.

Declaration

```
// C#
public async Task<Dictionary<string, object?>> GetAsync(Object key,
RecordRetrieveOptions? options = null, CancellationToken cancellationToken =
default);
```

Parameters

- `key`
The unique ID associated with the record to retrieve.
- `options`
Optional options for retrieving the record.
- `cancellationToken`
The cancellation token.

Return Value

`Task<Dictionary<string, object?>>` representing the record if found, or null if not found.

Implements

`Microsoft.Extensions.VectorData.VectorStoreCollection`

Exceptions

`VectorStoreException`: The command fails to execute for any reason other than the absence of a record.

Remarks

Unlike `OracleCollection`, `OracleDynamicCollection` uses `Object` as `TKey`, and `Dictionary<string, object?>` as `TRecord`. Please make sure to use the correct parameter types.

① See Also

- [RecordRetrievalOptions](#)

GetAsync(IEnumerable<Object>, RecordRetrieveOptions?, CancellationToken)

This method retrieves a batch of records from the vector store. It does not guarantee that the collection exists and returns null if the record is not found.

Declaration

```
// C#  
public async IEnumerable<Dictionary<string, object?>>  
GetAsync(IEnumerable<Object> keys, RecordRetrieveOptions? options = null,  
CancellationToken cancellationToken = default);
```

Parameters

- **key**
The unique IDs associated with the records to retrieve.
- **options**
Optional options for retrieving the records.
- **cancellationToken**
The cancellation token.

Return Value

`IAsyncEnumerable<Dictionary<string, object?>` object representing the result set of found records.

Implements

`Microsoft.Extensions.VectorData.VectorStoreCollection`

Exceptions

`VectorStoreException`: The command fails to execute for any reason other than the absence of a record.

Remarks

Unlike `OracleCollection`, `OracleDynamicCollection` uses `Object` as `TKey`, and `Dictionary<string, object?>` as `TRecord`. Please make sure to use the correct parameter types: `IEnumerable<Object>` for keys.

① See Also

- [RecordRetrievalOptions](#)

GetAsync(Expression<Func<Dictionary<string, object?>, bool>>,
int,<Dictionary<string, object?>>?, CancellationToken)

This method retrieves from the vector store records matching a filter expression. It does not guarantee the existence of the collection.

Declaration

```
// C#  
public async IEnumerable<Dictionary<string, object?>>  
GetAsync(Expression<Func<Dictionary<string, object?>, bool>> filter, int top,  
FilteredRecordRetrievalOptions<Dictionary<string, object?>>? options = null,  
CancellationToken cancellationToken = default);
```

Parameter

- filter
Predicate for filtering the records.
- top
Maximum number of results to return.
- options
Optional options for retrieving the records.
- cancellationToken
The cancellation token.

Return Value

IAsyncEnumerable<Dictionary<string, object?>> object representing the records matching the provided predicate.

Implements

Microsoft.Extensions.VectorData.VectorStoreCollection

Exceptions

VectorStoreException: The command fails to execute for any reason other than the absence of a record.

Remarks

The filter cannot compare the following unsupported columns types:

- BFile
- Blob
- Clob
- Json
- NClob
- Vector
- Vector_Binary

- Vector_Float32
- Vector_Float64
- Vector_Int8
- XmlType

Unlike OracleCollection, OracleDynamicCollection uses Object as TKey, and Dictionary<string, object?> as TRecord. Ensure to use the correct parameter types for the filter and options: Dictionary<string, object?>.

① See Also

- [FilteredRecordRetrievalOptions<TRecord> Class](#)

GetService

This method retrieves an object of a specified service type from the vector store.

Declaration

```
// C#  
public object? GetService(Type serviceType, object? serviceKey = null);
```

Parameters

- serviceType
The type of an object to retrieve. Supported ServiceType: VectorStoreCollectionMetaData, OracleCollection or OracleDataSource (.NET Core only).
- serviceKey
An optional key that can be used to identify the target service.

Return Value

An object of a specified service type, otherwise null.

Implements

Microsoft.Extensions.VectorData.VectorStoreCollection

SearchAsync

This method searches the vector store for records that are similar to input data.

Declaration

```
// C#  
public async IEnumerable<VectorSearchResult<Dictionary<string, object?>>> SearchAsync<TInput>(TInput searchValue, int top, VectorSearchOptions<Dictionary<string, object?>>? options = null, CancellationToken cancellationToken = default) where TInput : notnull;
```

Parameters

- **TInput**
The type of the input value to perform the similarity search.
- **searchValue**
Input value for similarity search. For more information, see the Remarks section.
- **top**
The maximum number of results to return.
- **options**
The options that control search behavior.
- **cancellationToken**
The cancellation token.

Return Value

`IAsyncEnumerable<VectorSearchResult<Dictionary<string, object?>>` records found by the vector search, including their result scores.

Implements

`Microsoft.Extensions.VectorData.IVectorSearchable`

Exceptions

- **VectorStoreException**: The command fails to execute for any reason other than the absence of a record.
- **NotSupportedException**: The database is not a 26ai or later database that does not have vector data type support.

Remarks

The types supported for the `searchValue` parameter vary based on the provider being used and the embedding generation configured.

A string or [DataContent](#) (for images, sound...) if an appropriate [IEmbeddingGenerator](#) has been configured that accepts that type as input. For example, register an [IEmbeddingGenerator](#) that accepts string as input in your dependency injection container, and then pass in a string argument to this method; the argument will be automatically passed to the [IEmbeddingGenerator](#) to generate the embedding and perform the search.

Some databases support generating embeddings at the database side. In this case, you can pass in a string or [DataContent](#) without configuring an [IEmbeddingGenerator](#) with [Microsoft.Extensions.VectorData..](#) The provider will simply send your argument to the database as-is for embedding generation.

Arbitrary .NET types can also be passed in as long as an appropriate [IEmbeddingGenerator](#) has been configured; for example, you can create your own [IEmbeddingGenerator](#) that accepts your own custom types as input, and uses another [IEmbeddingGenerator](#) to generate embedding from multiple properties.

For .NET types beyond string and [DataContent](#), you must use the generic [VectorStoreVectorProperty](#) in your record definition.

To work with embeddings directly, pass in the .NET data types that the connector accepts for Vector data type.

If you are using [IEmbeddingGenerator](#) directly in your code, that type returns an [Embedding](#) (for example, `Embedding`), which can also be passed in directly, as long as the provider supports the specific embedding type. However, consider registering your [IEmbeddingGenerator](#) with the provider instead and pass in the input type (for example, `string`).

Unlike `OracleCollection`, `OracleDynamicCollection` uses `Object` as `TKey`, and `Dictionary<string, object?>` as `TRecord`. Ensure to use the correct parameter types for options: `Dictionary<string, object?>`.

See Also

- [VectorSearchOptions<TRecord> Class](#)

UpsertAsync

`UpsertAsync` methods update or insert a record or a batch of records into the vector store. It does not guarantee the existence of the collection. If the record already exists, it is updated; if it does not exist, it is created.

Overload List

- [UpsertAsync\(Dictionary<string, object?>, CancellationToken\)](#)
This method updates or inserts a record into the vector store. It does not guarantee the existence of the collection. If the record already exists, it is updated; if it does not exist, it is created.
- [UpsertAsync\(IEnumerable<Dictionary<string, object?>>, CancellationToken\)](#)
This method updates or inserts a batch of records into the vector store. It does not guarantee the existence of the collection. If the record already exists, it is updated; if it does not exist, it is created.

UpsertAsync(Dictionary<string, object?>, CancellationToken)

This method updates or inserts a record into the vector store. It does not guarantee the existence of the collection. If the record already exists, it is updated; if it does not exist, it is created.

Declaration

```
// C#
public async Task UpsertAsync(Dictionary<string, object?> record,
    CancellationToken cancellationToken = default);
```

Parameters

- `record`
The record to update or insert.
- `cancellationToken`
The cancellation token.

Return Value

Task representing the asynchronous operation when the records are updated or inserted.

Implements

Microsoft.Extensions.VectorData.VectorStoreCollection

Exceptions

VectorStoreException: The command fails to execute for any reason other than the absence of a record.

Remarks

Unlike OracleCollection, OracleDynamicCollection uses Object as TKey, and Dictionary<string, object?> as TRecord. Please make sure to use the correct parameter types, Dictionary<string, object?>, for record.

UpsertAsync(IEnumerable<Dictionary<string, object?>>, CancellationToken)

This method updates or inserts a batch of records into the vector store. It does not guarantee the existence of the collection. If the record already exists, it is updated; if it does not exist, it is created.

Declaration

```
// C#  
public async Task UpsertAsync(IEnumerable<Dictionary<string, object?>>  
records, CancellationToken cancellationToken = default);
```

Parameters

- records
The records to update or insert.
- cancellationToken
The cancellation token.

Return Value

Task representing the asynchronous operation when the records are updated or inserted.

Implements

Microsoft.Extensions.VectorData.VectorStoreCollection

Exceptions

VectorStoreException: The command fails to execute for any reason other than the absence of a record.

Remarks

All records are upserted when succeed or none when fails.

Unlike OracleCollection, OracleDynamicCollection uses Object as TKey, and Dictionary<string, object?> as TRecord. Please make sure to use the correct parameter types, IEnumerable<Dictionary<string, object?>> for records.

OracleServiceCollectionExtensions Class

OracleServiceCollectionExtensions class is for extension methods to register OracleVectorStore or OracleCollection instances on an IServiceCollection. It extends the .NET IServiceCollection class through dependency injection to provide the application a service to register either OracleVectorStore or OracleCollection services.

Class Inheritance

System.Object
 Microsoft.Extensions.DependencyInjection.OracleServiceCollectionExtensions

Declaration

```
// C#  
public static class OracleServiceCollectionExtensions
```

Thread Safety

All public methods are thread-safe, unless stated otherwise.

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

See Also

- [OracleServiceCollectionExtensions Static Methods](#)
- [IServiceCollection Interface](#)

OracleServiceCollectionExtensions Static Methods

OracleServiceCollectionExtensions static methods are listed in [Table 21-6](#).

Table 21-6 OracleServiceCollectionExtensions Static Methods

Public Method	Description
AddKeyedOracleCollection	Registers a keyed OracleCollection as VectorStoreCollection, with the specified connection string and service lifetime.

Table 21-6 (Cont.) OracleServiceCollectionExtensions Static Methods

Public Method	Description
AddKeyedOracleVectorStore	Registers a keyed OracleVectorStore as VectorStore, with the specified connection string and service lifetime.
AddOracleCollection	Registers an OracleCollection as VectorStoreCollection, with the specified connection string and service lifetime.
AddOracleVectorStore	Registers an OracleVectorStore as VectorStore, with the specified connection string and service lifetime.

See Also

- [OracleServiceCollectionExtensions Class](#)

AddKeyedOracleCollection

AddKeyedOracleCollection methods register a keyed OracleCollection as VectorStoreCollection.

Overload List

- [AddKeyedOracleCollection<TKey, TRecord>\(IServiceCollection, object?, string, Func<IServiceProvider, string>, Func<IServiceProvider, OracleVectorCollectionOptions>?, ServiceLifetime \)](#)
This method registers a keyed OracleCollection as VectorStoreCollection, with the specified connection string provider and service lifetime.
- [AddKeyedOracleCollection<TKey, TRecord>\(IServiceCollection, object?, string, string, OracleCollectionOptions?, ServiceLifetime\)](#)
This method registers a keyed OracleCollection as VectorStoreCollection, with the specified connection string and service lifetime.

AddKeyedOracleCollection<TKey, TRecord>(IServiceCollection, object?, string, Func<IServiceProvider, string>, Func<IServiceProvider, OracleVectorCollectionOptions>?, ServiceLifetime)

This method registers a keyed OracleCollection<TKey, TRecord> as VectorStoreCollection, with the specified connection string provider and service lifetime.

Declaration

```
// C#
public static IServiceCollection AddKeyedOracleCollection<TKey, TRecord>(this
IServiceCollection services, object? serviceKey, string name,
Func<IServiceProvider, string> connectionStringProvider,
Func<IServiceProvider, OracleVectorCollectionOptions>? optionsProvider =
```



```
null, ServiceLifetime lifetime = ServiceLifetime.Singleton) where TKey :  
notnull where TRecord : class
```

Parameters

- **TKey**
The key type of `OracleCollection<TKey, TRecord>`.
- **TRecord**
The record type of `OracleCollection<TKey, TRecord>`.
- **services**
The `IServiceCollection` to register the `VectorStore` on.
- **serviceKey**
The key to associate with the vector collection.
- **name**
The name of the collection.
- **connectionStringProvider**
The connection string provider.
- **optionsProvider**
The optional options provider to further configure the vector store.
- **lifetime**
The service lifetime for the store. Defaults to `ServiceLifetime.Singleton`.

Return Value

Returns the service collection.

Exceptions

`InvalidArgumentException`:

- The provided `connectionStringProvider` or `services` parameter is null.
- The provided `name` parameter is null or empty.

AddKeyedOracleCollection<TKey, TRecord>(IServiceCollection, object?, string, string, OracleCollectionOptions?, ServiceLifetime)

This method registers a keyed `OracleCollection<TKey, TRecord>` as `VectorStoreCollection`, with the specified connection string and service lifetime.

Declaration

```
// C#  
public static IServiceCollection AddKeyedOracleCollection<TKey, TRecord>(this  
IServiceCollection services, object? serviceKey, string name, string  
connectionString, OracleCollectionOptions? options = default, ServiceLifetime  
lifetime = ServiceLifetime.Singleton) where TKey : notnull where TRecord :  
class
```

Parameters

- **services**

The IServiceCollection to register the VectorStore on.

- **serviceKey**
The key to associate with the vector collection.
- **name**
The name of the collection.
- **connectionString**
The connection string.
- **optionsProvider**
The optional options provider to further configure the vector store.
- **lifetime**
The service lifetime for the store. Defaults to ServiceLifetime.Singleton.

Return Value

Returns the service collection.

AddKeyedOracleVectorStore

AddKeyedOracleVectorStore methods register a keyed OracleVectorStore as VectorStore.

Overload List

- [AddKeyedOracleVectorStore\(IServiceCollection, object?, Func<IServiceProvider, string>, Func<IServiceProvider, OracleVectorStoreOptions>?, ServiceLifetime\)](#)
This method registers a keyed OracleVectorStore as VectorStore, with the specified connection string provider and service lifetime.
- [AddKeyedOracleVectorStore\(IServiceCollection, object?, string, OracleVectorStoreOptions?, ServiceLifetime\)](#)
This method registers a keyed OracleVectorStore as VectorStore, with the specified connection string and service lifetime.

AddKeyedOracleVectorStore(IServiceCollection, object?, Func<IServiceProvider, string>, Func<IServiceProvider, OracleVectorStoreOptions>?, ServiceLifetime)

This method registers a keyed OracleVectorStore as VectorStore, with the specified connection string provider and service lifetime.

Declaration

```
// C#
public static IServiceCollection AddKeyedOracleVectorStore(this
IServiceCollection services, object? serviceKey, Func<IServiceProvider,
string> connectionStringProvider, Func<IServiceProvider,
OracleVectorStoreOptions>? optionsProvider = null, ServiceLifetime lifetime =
ServiceLifeTime.Singleton)
```

Parameters

- **services**
The IServiceCollection to register the VectorStore on.
- **serviceKey**

The key to associate with the vector store.

- `connectionStringProvider`
The connection string provider.
- `optionsProvider`
The optional options provider to further configure the vector store.
- `lifetime`
The service lifetime for the store. Defaults to `ServiceLifetime.Singleton`.

Return Value

Returns the service collection.

Exceptions

`InvalidArgumentException`: The provided `connectionStringProvider` or `services` parameters is null.

AddKeyedOracleVectorStore(IServiceCollection, object?, string, OracleVectorStoreOptions?, ServiceLifetime)

This method registers a keyed `OracleVectorStore` as `VectorStore`, with the specified connection string and service lifetime.

Declaration

```
// C#
public static IServiceCollection AddKeyedOracleVectorStore(this
IServiceCollection services, object? serviceKey, string
connectionString, OracleVectorStoreOptions? options = null, ServiceLifetime
lifetime = ServiceLifetime.Singleton)
```

Parameters

- `services`
The `IServiceCollection` to register the `VectorStore` on.
- `serviceKey`
The key to associate with the vector store.
- `connectionString`
The connection string.
- `options`
Options to further configure the vector store.
- `lifetime`
The service lifetime for the store. Defaults to `ServiceLifetime.Singleton`.

Return Value

Returns the service collection.

AddOracleCollection

`AddOracleCollection` methods register an `OracleCollection` as `VectorStoreCollection`.

Overload List

- [AddOracleCollection<TKey, TRecord>\(IServiceCollection, string, Func<IServiceProvider, string>, Func<IServiceProvider, OracleCollectionsOptions>?, ServiceLifetime\)](#)
This method registers an OracleCollection as VectorStoreCollection, with the specified connection string and service lifetime.
- [AddOracleCollection<TKey, TRecord>\(IServiceCollection, string, string, OracleCollectionOptions?, ServiceLifetime\)](#)
This method registers an OracleCollection as VectorStoreCollection, with the specified connection string and service lifetime.
- [AddOracleCollection<TKey, TRecord>\(this IServiceCollection, string, OracleCollectionOptions?, ServiceLifetime\)](#)
This method registers an OracleCollection as VectorStoreCollection, with the OracleDataSource retrieved from the IServiceCollection.

AddOracleCollection<TKey, TRecord>(IServiceCollection, string, Func<IServiceProvider, string>, Func<IServiceProvider, OracleCollectionsOptions>?, ServiceLifetime)

This method registers an OracleCollection as VectorStoreCollection, with the specified connection string provider and service lifetime.

Declaration

```
// C#
public static IServiceCollection AddOracleCollection<TKey, TRecord>(this
IServiceCollection services, string name, Func<IServiceProvider, string>
connectionStringProvider, Func<IServiceProvider, OracleCollectionsOptions>?
optionsProvider = null, ServiceLifetime lifetime = ServiceLifeTime.Singleton)
where TKey : notnull where TRecord : class
```

Parameters

- **services**
The IServiceCollection to register the VectorStore on.
- **name**
The name of the collection.
- **connectionStringProvider**
The connection string provider.
- **optionsProvider**
The optional options provider to further configure the vector store.
- **lifetime**
The service lifetime for the store. Defaults to ServiceLifetime.Singleton.

Return Value

Returns the service collection.

Exceptions

InvalidArgumentException:

- The provided connectionStringProvider or services parameter is null.
- The provided name parameter is null or empty.

AddOracleCollection<TKey, TRecord>(IServiceCollection, string, string, OracleCollectionOptions?, ServiceLifetime)

This method registers an OracleCollection as VectorStoreCollection, with the specified connection string and service lifetime.

Declaration

```
// C#
public static IServiceCollection AddOracleCollection<TKey, TRecord>(this
IServiceCollection services, string name, string
connectionString, OracleCollectionOptions? options = default, ServiceLifetime
lifetime = ServiceLifetime.Singleton) where TKey : notnull where TRecord : class
```

Parameters

- services
The IServiceCollection to register the VectorStore on.
- name
The name of the collection.
- connectionString
The connection string.
- options
Options to further configure the vector store.
- lifetime
The service lifetime for the store. Defaults to ServiceLifetime.Singleton.

Return Value

Returns the service collection.

Exceptions

InvalidArgumentException

- The provided services parameter is null.
- The provided connectionString or name parameter is null or empty.

AddOracleCollection<TKey, TRecord>(this IServiceCollection, string, OracleCollectionOptions?, ServiceLifetime)

This method registers an OracleCollection as VectorStoreCollection, with the OracleDataSource retrieved from the IServiceCollection.

Declaration

```
// C#
public static IServiceCollection AddOracleCollection(this IServiceCollection
```

```
services, string name, OracleCollectionsOptions? options = null,
ServiceLifetime lifetime = ServiceLifeTime.Singleton) where TKey : notnull
where TRecord : class;
```

Parameters

- **services**
The IServiceCollection to register the VectorStore on.
- **name**
The name of the collection.
- **options**
Options to further configure the vector store.
- **lifetime**
The service lifetime for the store. Defaults to ServiceLifetime.Singleton.

Return Value

Returns the service collection.

Exceptions

InvalidArgumentException:

- The provided services parameter is null.
- The provided name parameter is null or empty.

InvalidOperationException: The service collection does not contain an OracleDataSource.

AddOracleVectorStore

AddOracleVectorStore methods register an OracleVectorStore as VectorStore.

Overload List

- [AddOracleVectorStore\(IServiceCollection, Func<IServiceProvider, string>, Func<IServiceProvider, OracleVectorStoreOptions>?, ServiceLifetime\)](#)
This method registers an OracleVectorStore as VectorStore, with the specified connection string and service lifetime.
- [AddOracleVectorStore\(IServiceCollection, string, OracleVectorStoreOptions?, ServiceLifetime\)](#)
This method registers an OracleVectorStore as VectorStore, with the specified connection string and service lifetime.
- [AddOracleVectorStore\(this IServiceCollection, OracleVectorStoreOptions?, ServiceLifetime\)](#)
This method registers an OracleVectorStore as VectorStore, with the OracleDataSource retrieved from the IServiceCollection.

AddOracleVectorStore(this IServiceCollection, OracleVectorStoreOptions?, ServiceLifetime)

This method registers an OracleVectorStore as VectorStore, with the OracleDataSource retrieved from the IServiceCollection.

Declaration

```
// C#  
public static IServiceCollection AddOracleVectorStore(this IServiceCollection  
services, OracleVectorStoreOptions? options = null, ServiceLifetime lifetime  
= ServiceLifetime.Singleton);
```

Parameters

- **services**
The IServiceCollection to register the VectorStore on.
- **options**
Options to further configure the vector store.
- **lifetime**
The service lifetime for the store. Defaults to ServiceLifetime.Singleton.

Return Value

Returns the service collection.

Exceptions

InvalidArgumentException: The provided services parameter is null.

InvalidOperationException: The service collection does not contain an OracleDataSource.

AddOracleVectorStore(IServiceCollection, string, OracleVectorStoreOptions?, ServiceLifetime)

This method registers an OracleVectorStore as VectorStore, with the specified connection string and service lifetime.

Declaration

```
// C#  
public static IServiceCollection AddOracleVectorStore(this IServiceCollection  
services, string connectionString, OracleVectorStoreOptions? options =  
null, ServiceLifetime lifetime = ServiceLifetime.Singleton)
```

Parameters

- **services**
The IServiceCollection to register the VectorStore on.
- **connectionString**
The connection string.
- **options**
Options to further configure the vector store.
- **lifetime**
The service lifetime for the store. Defaults to ServiceLifetime.Singleton.

Return Value

Returns the service collection.

Exceptions

InvalidArgumentException

- The provided services parameter is null.
- The provided connectionString parameter is null or empty string.

AddOracleVectorStore(IServiceCollection, Func<IServiceProvider, string>, Func<IServiceProvider, OracleVectorStoreOptions>?, ServiceLifetime)

This method registers an `OracleVectorStore` as `VectorStore`, with the specified connection string provider and service lifetime.

Declaration

```
// C#
public static IServiceCollection AddOracleVectorStore(this IServiceCollection
services, Func<IServiceProvider, string> connectionStringProvider,
Func<IServiceProvider, OracleVectorStoreOptions>? optionsProvider = null,
ServiceLifetime lifetime = ServiceLifetime.Singleton)
```

Parameters

- `services`
The `IServiceCollection` to register the `VectorStore` on.
- `connectionStringProvider`
The connection string provider.
- `optionsProvider`
Options provider to further configure the vector store.
- `lifetime`
The service lifetime for the store. Defaults to `ServiceLifetime.Singleton`.

Return Value

Returns the service collection.

Exceptions

InvalidOperationException: The provided `connectionStringProvider` or `services` parameter is null.

OracleVectorStore Class

An `OracleVectorStore` object implements the `Microsoft.Extensions.VectorData.VectorStore` interface. It represents a database connection and provides access to a database. It is primarily used for creating, deleting and getting a collection (table).

Class Inheritance

System.Object
 Microsoft.Extensions.VectorData.VectorStore
 Oracle.VectorData.OracleVectorStore

Declaration

```
// C#  
public sealed class OracleVectorStore :  
Microsoft.Extensions.VectorData.VectorStore
```

Thread Safety

All public methods are thread-safe, unless stated otherwise.

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

See Also

- [OracleVectorStore Constructors](#)
- [OracleVectorStore Public Methods](#)

OracleVectorStore Constructors

OracleVectorStore constructors instantiate a new instance of OracleVectorStore class using either the provided data source or a connection string.

Overload List:

- [OracleVectorStore\(OracleDataSource, OracleVectorStoreOptions?\)](#)
This constructor creates an instance of the OracleVectorStore class using the provided data source. It is applicable only to .NET Core 8.0 or above versions.
- [OracleVectorStore\(string connectionString, OracleVectorStoreOptions?\)](#)
This constructor creates an instance of the OracleVectorStore class using the provided database connection string.

① See Also

- [OracleVectorStore Class](#)
- [OracleVectorStore Public Methods](#)

OracleVectorStore(OracleDataSource, OracleVectorStoreOptions?)

Instantiates a new instance of `OracleVectorStore` class using the provided data source.

Declaration

```
// C#  
OracleVectorStore(OracleDataSource dataSource, OracleVectorStoreOptions?  
options = default)
```

Parameters

- *dataSource*
The data source that can be used to obtain open database connections.
- *options*
Optional vector store configuration options.

① Note

When these options are specified, the collections created within the vector store inherit these options.

Remarks

The vector store is created based on the following connector defined values in `Microsoft.Extensions.VectorData.VectorStore.VectorStoreMetaData` class:

- `VectorStoreSystemName`: "Oracle"
- `VectorStoreName`: The name of the database that the vector store is connected to. This value is auto-populated from `OracleConnection.DatabaseName`.

Exceptions

`ArgumentNullException`: The provided `dataSource` parameter is null.

① See Also

- [OracleDataSource Class](#)

OracleVectorStore(string connectionString, OracleVectorStoreOptions?)

Instantiates a new instance of `OracleVectorStore` class using the provided database connection string.

Declaration

```
// C#  
OracleVectorStore(string connectionString, OracleVectorStoreOptions? options  
= default)
```

Parameters

- *connectionString*
The connection string which can be used to obtain database connections.
- *options*
Optional vector store configuration options.

Note

When these options are specified, the collections created within the vector store inherit these options.

Remarks

The vector store is created based on the following connector defined values in `Microsoft.Extensions.VectorData.VectorStore.VectorStoreMetaData`:

- `VectorStoreSystemName`: "Oracle"
- `VectorStoreName`: The database name that the vector store is connected to. This value is auto-populated from `OracleConnection.DatabaseName`.

Exceptions

- `ArgumentNullException`: The provided `connectionString` parameter is null.
- `ArgumentException`: The provided `connectionString` parameter is empty or whitespace.

See Also

- [OracleConnection.ConnectionString](#)

OracleVectorStore Public Methods

OracleVectorStore public methods are listed in [Table 21-7](#).

Table 21-7 OracleVectorStore Public Methods

Public Method	Description
CollectionExistsAsync	Checks if a collection exists in the vector store.
Dispose	Releases any resources or memory held by the object.
EnsureCollectionDeletedAsync	Deletes a specified collection.

Table 21-7 (Cont.) OracleVectorStore Public Methods

Public Method	Description
GetCollection(string, VectorStoreCollectionDefinition?)	Creates a vector store collection using its name.
GetDynamicCollection(string, VectorStoreCollectionDefinition)	Creates a vector store collection using its name.
GetService	Retrieves an object of a specified service type from the vector store.
ListCollectionNamesAsync	Retrieves the names of all the collections in the vector store.

See Also

- [OracleVectorStore Class](#)
- [OracleVectorStore Constructors](#)

CollectionExistsAsync

This method checks if a collection exists in the vector store.

Declaration

```
// C#  
public Task<bool> CollectionExistsAsync(string name, CancellationToken  
cancellationTok = default)
```

Parameters

- `name`
The name of the collection.
- `cancellationTok`
The cancellation token.

Return Value

A task representing the asynchronous operation. The task result contains a boolean indicating whether the collection exists in the database. True if exists, else false.

Implements

Microsoft.Extensions.VectorData.VectorStore

Dispose

This method releases any resources or memory allocated by the object.

Declaration

```
// C#  
public void Dispose()
```

Implements

System.IDisposable

Thread Safety

This method is not thread-safe.

EnsureCollectionDeletedAsync

This method deletes a specified collection.

Declaration

```
// C#  
public Task EnsureCollectionDeletedAsync(string name, CancellationToken  
cancellationTok = default)
```

Parameters

- **name**
The name of the collection.
- **cancellationTok**
The cancellation token.

Return Value

A task representing the asynchronous operation that completes when the collection has been deleted from the database.

Implements

Microsoft.Extensions.VectorData.VectorStore

GetCollection(string, VectorStoreCollectionDefinition?)

This method creates a vector store collection using its name.

Declaration

```
// C#  
public VectorStoreCollection<TKey, TRecord> GetCollection(string  
collectionName, VectorStoreCollectionDefinition? definition = null)where  
TKey : notnullwhere TRecord : notnull
```

Parameters

- **TKey**
The data type of the record key.

- **TRecord**
The record data model to use for adding, updating, and retrieving data from the collection.
- **collectionName**
The name of the collection.
- **definition**
The schema of the record type.

Return Value

A new `OracleCollection<TKey, TRecord>` instance for managing records in the collection.

Implements

`Microsoft.Extensions.VectorData.VectorStore`

Exceptions

`ArgumentException`:

- The provided `collectionName` parameter is null, an empty string or contains white spaces only.
- The provided `collectionName` is not compliant with Oracle name standard.
- The provided `TKey` and `TRecord` cannot be mapped to an Oracle type based on the default/custom type mapping.

Remarks

To successfully request a collection, either must be annotated with attributes that define the schema of the record type, or must be provided.

The `OracleCollection` instance is created for application to operate on a collection. However, the collection is not created in the database. To create a collection in the database, use `OracleCollection.EnsureCollectionExistsAsync()`.

See Also

`Microsoft.Extensions.VectorData.VectorStoreKeyAttribute`,
`VectorStoreDataAttribute`, `VectorStoreVectorAttribute` classes.

GetDynamicCollection(string, VectorStoreCollectionDefinition)

This method creates a vector store collection using its name.

Declaration

```
// C#
public VectorStoreCollection<object, Dictionary<string, object?>>
GetDynamicCollection(string collectionName, VectorStoreRecordDefinition
definition)
```

Parameters

- `collectionName`

Name of the collection to retrieve.

- **definition**
The schema of the record type.

Return Value

A new `OracleDynamicCollection` instance for managing the records in the collection.

Implements

`Microsoft.Extensions.VectorData.VectorStore`

Exceptions

`ArgumentException`:

- The provided `collectionName` parameter is null, an empty string or contains white spaces only.
- The provided `collectionName` is not compliant with Oracle name standard.

Remarks

The `OracleDynamicCollection` instance is created for application to operate on a collection. However, the collection is not created in the database. To create a collection in the database, use `OracleCollection.EnsureCollectionExistsAsync()`.

The provided collection name is treated as case-sensitive and should not be a fully qualified name(like `.`) as the Schema is referred based on the `OracleVectorStoreOptions`.

See Also

`Microsoft.Extensions.VectorData.VectorStoreCollectionDefinition` class.

GetService

This method retrieves an object of a specified service type from the vector store.

Declaration

```
// C#  
public object? GetService(Type serviceType, object? serviceKey = null);
```

Parameters

- **serviceType**
The type of an object to retrieve. Supported `ServiceType`: `VectorStoreMetaData`, `OracleVectorStore` or `OracleDataSource` (.NET Core only).
- **serviceKey**
An optional key that can be used to identify the target service.

Return Value

An object of a specified service type, otherwise null.

Implements

Microsoft.Extensions.VectorData.VectorStore

ListCollectionNamesAsync

This method retrieves the names of all the collections in the vector store.

Declaration

```
// C#
public async IEnumerable<string>
ListCollectionNamesAsync(CancellationToken cancellationToken = default)
```

Parameters

- `cancellationToken`
The cancellation token.

Return Value

The list of all the collection names in the vector store.

Implements

Microsoft.Extensions.VectorData.VectorStore

OracleVectorStoreOptions Class

An `OracleVectorStoreOptions` object encapsulates configuration options for creating and managing an instance of the `OracleVectorStore` class. It lets you to specify the database schema and the default embedding generator to use for vector operations.

Class Inheritance

System.Object
 Oracle.VectorData.OracleVectorStoreOptions

Declaration

```
// C#
public sealed class OracleVectorStoreOptions
```

Requirements

Provider	ODP.NET, Managed Driver	ODP.NET Core
Assembly	Oracle.ManagedDataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.ManagedDataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	See System Requirements	-
.NET (Core)	-	See System Requirements

① See Also

- [OracleVectorStoreOptions Constructor](#)
- [OracleVectorStoreOptions Properties](#)

OracleVectorStoreOptions Constructor

Instantiates a new instance of the OracleVectorStoreOptions class.

Declaration

```
// C#
public OracleVectorStoreOptions()
```

① See Also

- [OracleVectorStoreOptions Class](#)
- [OracleVectorStoreOptions Properties](#)

OracleVectorStoreOptions Properties

OracleVectorStoreOptions properties are listed in [Table 21-8](#).

Table 21-8 OracleVectorStoreOptions Properties

Property	Description
EmbeddingGenerator	This property specifies the default embedding generator to use for generating vector embeddings when working with the vector store.
Schema	This property specifies the database schema name under which collections are created by default.

① See Also

- [OracleVectorStoreOptions Class](#)
- [OracleVectorStoreOptions Constructor](#)

EmbeddingGenerator

This property specifies the default embedding generator to use for generating vector embeddings when working with the vector store.

Declaration

```
// C#  
public IEmbeddingGenerator? EmbeddingGenerator { get; set; }
```

Property Value

The default embedding generator to use for generating vector embeddings when working with the vector store.

Remarks

When embedding is specified, the collections under this vector store uses the `EmbeddingGenerator` to generate embeddings. The default property value is null. When the embedding generator is provided, the vector store automatically generates the embedding during both upsert and search operations, eliminating the need for the application to generate and provide the embedding.

Schema

This property specifies the database schema name under which collections are created.

Declaration

```
// C#  
public string? Schema { get; set; }
```

Property Value

The string specifying the name of the database schema.

Remarks

When `Schema` is specified, the collections under this vector store are created under the schema. The default schema value is null. When null, the collections are created under the connected user's schema.

A

Oracle Schema Collections

ODP.NET provides standard metadata collections as well as various Oracle database-specific metadata collections that can be retrieved through the `OracleConnection.GetSchema` API.

See Also

- ["Schema Discovery"](#)
- ["GetSchema"](#)

This appendix contains the following topics:

- [Common Schema Collections](#)
- [ODP.NET-Specific Schema Collection](#)

Common Schema Collections

The common schema collections are available for all .NET Framework managed providers. ODP.NET supports the same common schema collections.

- [MetaDataCollections](#)
- [DataSourceInformation](#)
- [DataTypes](#)
- [Restrictions](#)
- [ReservedWords](#)

See Also

"Common Schema Collections" in the [MSDN Library](#)

MetaDataCollections

[Table A-1](#) is a list of metadata collections that is available from the data source, such as tables, columns, indexes, and stored procedures.

Table A-1 MetaDataCollections

Column Name	Data Type	Description
CollectionName	string	The name of the collection passed to the <code>GetSchema</code> method for retrieval.

Table A-1 (Cont.) MetaDataCollections

Column Name	Data Type	Description
NumberOfRestrictions	int	Number of restrictions specified for the named collection.
NumberOfIdentifierParts	int	Number of parts in the composite identifier/database object name.

DataSourceInformation

[Table A-2](#) lists DataSourceInformation information which may include these columns and possibly others.

Table A-2 DataSource Information

Columns	Data Type	Description
CompositeIdentifierSeparatorPattern	string	Separator for multipart names: @ \ .
DataSourceProductName	string	Database name: Oracle
DataSourceProductVersion	string	Database version. Note that this is the version of the database instance currently being accessed by DbConnection.
DataSourceProductVersionNormalized	string	A normalized DataSource version for easier comparison between different versions. For example: DataSource Version: 10.2.0.1.0 Normalized DataSource Version: 10.02.00.01.00
GroupByBehavior	GroupByBehavior	An enumeration that indicates the relationship between the columns in a GROUP BY clause and the non-aggregated columns in a select list.
IdentifierPattern	string	Format for a valid identifier.
IdentifierCase	IdentifierCase	An enumeration that specifies whether or not to treat non-quoted identifiers as case sensitive.
OrderByColumnsInSelect	bool	A boolean that indicates whether or not the select list must contain the columns in an ORDER BY clause.
ParameterMarkerFormat	string	A string indicating whether or not parameter markers begin with a special character.
ParameterMarkerPattern	string	The format of a parameter marker.
ParameterNameMaxLength	int	Maximum length of a parameter.
ParameterNamePattern	string	The format for a valid parameter name.
QuotedIdentifierPattern	string	The format of a quoted identifier.
QuotedIdentifierCase	IdentifierCase	An enumeration that specifies whether or not to treat quote identifiers as case sensitive.
StringLiteralPattern	string	The format for a string literal.

Table A-2 (Cont.) DataSource Information

Columns	Data Type	Description
SupportedJoinOperators	SupportedJoinOperators	An enumeration indicating the types of SQL join statements supported by the data source.

DataTypes

[Table A-3](#) lists DataTypes Collection information which may include these columns and possibly others.

Note

As an example, the description column includes complete information for the `TIMESTAMP WITH LOCAL TIME ZONE` data type.

Table A-3 Data Types

ColumnName	Data Type	Description
TypeName	string	The provider-specific data type name. Example: <code>TIMESTAMP WITH LOCAL TIME ZONE</code>
ProviderDbType	int	The provider-specific type value. Example: 124
ColumnSize	long	The length of a non-numeric column or parameter. Example: 27
CreateFormat	string	A format string that indicates how to add this column to a DDL statement. Example: <code>TIMESTAMP({0}) WITH LOCAL TIME ZONE</code>
CreateParameters	string	The parameters specified to create a column of this data type. Example: 8
DataType	string	The .NET type for the data type. Example: <code>System.DateTime</code>
IsAutoIncrementable	bool	A boolean value that indicates whether or not this data type can be auto-incremented. Example: false

Table A-3 (Cont.) Data Types

ColumnName	Data Type	Description
IsBestMatch	bool	A boolean value that indicates whether or not this data type is the best match to values in the DataType column. Example: false
IsCaseSensitive	bool	A boolean value that indicates whether or not this data type is case-sensitive. Example: false
IsFixedLength	bool	A boolean value that indicates whether or not this data type has a fixed length. Example: true
IsFixedPrecisionScale	bool	A boolean value that indicates whether or not this data type has a fixed precision and scale. Example: false
IsLong	bool	A boolean value that indicates whether or not this data type contains very long data. Example: false
IsNullable	bool	A boolean value that indicates whether or not this data type is nullable. Example: true
IsSearchable	bool	A boolean value that indicates whether or not the data type can be used in a WHERE clause with any operator, except the LIKE predicate. Example: true
IsSearchableWithLike	bool	A boolean value that indicates whether or not this data type can be used with the LIKE predicate. Example: false
IsUnsigned	bool	A boolean value that indicates whether or not the data type is unsigned.
MaximumScale	short	The maximum number of digits allowed to the right of the decimal point.
MinimumScale	short	The minimum number of digits allowed to the right of the decimal point.

Table A-3 (Cont.) Data Types

ColumnName	Data Type	Description
IsConcurrencyType	bool	A boolean value that indicates whether or not the database updates the data type every time the row is changed and the value of the column differs from all previous values. Example: false
MinimumVersion	String	The earliest version of the database that can be used. Example: 09.00.00.00.00
IsLiteralSupported	bool	A boolean value that indicates whether or not the data type can be expressed as a literal. Example: true
LiteralPrefix	string	The prefix of a specified literal. Example: TO_TIMESTAMP_TZ(' '
LiteralSuffix	string	The suffix of a specified literal. Example: ', 'YYYY-MM-DD HH24:MI:SS.FF')

Restrictions

[Table A-4](#) lists Restrictions, including the following columns.

Table A-4 Restrictions

ColumnName	Data Type	Description
CollectionName	string	The collection that the restrictions apply to.
RestrictionName	string	The restriction name.
RestrictionNumber	int	A number that indicates the location of the restriction.

ReservedWords

The ReservedWords collection exposes information about the words that are reserved by the database currently connected to ODP.NET.

[Table A-5](#) lists the ReservedWords Collection.

Table A-5 ReservedWords

ColumnName	Data Type	Description
ReservedWord	string	Provider-specific reserved words

ODP.NET-Specific Schema Collection

Oracle Data Provider for .NET supports both the common schema collections described previously and the following Oracle-specific schema collections:

- [Tables](#)
- [Columns](#)
- [Views](#)
- [XMLSchema](#)
- [Users](#)
- [Synonyms](#)
- [Sequences](#)
- [Functions](#)
- [Procedures](#)
- [ProcedureParameters](#)
- [Arguments](#)
- [Packages](#)
- [PackageBodies](#)
- [JavaClasses](#)
- [Indexes](#)
- [IndexColumns](#)
- [PrimaryKeys](#)
- [ForeignKeys](#)
- [ForeignKeyColumns](#)
- [UniqueKeys](#)

Tables

[Table A-6](#) lists the column name, data type, and description of the Tables Schema Schema Collection.

Table A-6 Tables

Column Name	Data Type	Description
OWNER	String	Owner of the Table.
TABLE_NAME	String	Name of the Table.
TYPE	String	Type of Table, for example, System or User.

Columns

[Table A-7](#) lists the column name, data type, and description of the Columns Schema Collection .

Table A-7 Columns

ColumnName	Data Type	Description
OWNER	String	Owner of the table or view.
TABLE_NAME	String	Name of the table or view.
COLUMN_NAME	String	Name of the column.
ID	Decimal	Sequence number of the column as created.
DATATYPE	String	Data type of the column.
LENGTH	Decimal	Length of the column in bytes.
PRECISION	Decimal	Decimal precision for NUMBER data type; binary precision for FLOAT data type, null for all other data types.
Scale	Decimal	Digits to right of decimal point in a number.
NULLABLE	String	Specifies whether or not a column allows NULLs.
CHAR_USED	String	Indicates whether the column uses BYTE length semantics (B) or CHAR length semantics (C).
LengthInChars	Decimal	Length of the column in characters. This value only applies to CHAR, VARCHAR2, NCHAR, and NVARCHAR2.

Views

[Table A-8](#) lists the column name, data type, and description of the Views Schema Collection.

Table A-8 Views

Column Name	Data Type	Description
OWNER	String	Owner of the view.
VIEW_NAME	String	Name of the view.
TEXT_LENGTH	Decimal	Length of the view text.
TEXT	String	View text.
TYPE_TEXT_LENGTH	Decimal	Length of the type clause of the typed view.
TYPE_TEXT	String	Type clause of the typed view.
OID_TEXT_LENGTH	Decimal	Length of the WITH OID clause of the typed view.
OID_TEXT	String	WITH OID clause of the typed view.
VIEW_TYPE_OWNER	String	Owner of the view type if the view is a typed view.
VIEW_TYPE	String	Type of the view if the view is a typed view.
SUPERVIEW_NAME	String	Name of the superview. (Oracle9i or later)

XMLSchema

[Table A-9](#) lists the column name, data type and description of the XMLSchema Schema Collection.

Note

This collection is only available with Oracle Database 10g and later.

Table A-9 XMLSchema

Column Name	Data Type	Description
OWNER	String	Owner of the XML schema.
SCHEMA_URL	String	Schema URL of the XML schema.
LOCAL	String	Indicates whether the XML schema is local (YES) or global (NO).
SCHEMA	String	XML schema document.
INT_OBJNAME	String	Internal database object name for the schema.
QUAL_SCHEMA_URL	String	Fully qualified schema URL.
HIER_TYPE	String	Hierarchy type for the schema.

Users

[Table A-10](#) lists the column name, data type and description of the Users Schema Collection.

Table A-10 Users

Column Name	Data Type	Description
NAME	String	Name of the user.
ID	Decimal	ID number of the user.
CREATEDATE	DateTime	User creation date.

Synonyms

[Table A-11](#) lists the column name, data type and description of the Synonyms Schema Collection.

Table A-11 Synonyms

Column Name	Data Type	Description
OWNER	String	Owner of the synonym.
SYNONYM_NAME	String	Name of the synonym.
TABLE_OWNER	String	Owner of the object referenced by the synonym. Although the column is called TABLE_OWNER, the object owned is not necessarily a table. It can be any general object such as a view, sequence, stored procedure, synonym, and so on.

Table A-11 (Cont.) Synonyms

Column Name	Data Type	Description
TABLE_NAME	String	Name of the object referenced by the synonym. Although the column is called TABLE_NAME, the object does not necessarily have to be a table. It can be any general object such as a view, sequence, stored procedure, synonym, and so on.
DB_LINK	String	Name of the database link referenced, if any.

Sequences

[Table A-12](#) lists the column name, data type, and description of the Sequences Schema Collection.

Table A-12 Sequences

Column Name	Data Type	Description
SEQUENCE_OWNER	String	Name of the owner of the sequence.
SEQUENCE_NAME	String	Sequence name.
MIN_VALUE	Decimal	Minimum value of the sequence.
MAX_VALUE	Decimal	Maximum value of the sequence.
INCREMENT_BY	Decimal	Value by which sequence is incremented.
CYCLE_FLAG	String	Indicates if sequence wraps around on reaching limit.
ORDER_FLAG	String	Indicates if sequence numbers are generated in order.
CACHE_SIZE	Decimal	Number of sequence numbers to cache.
LAST_NUMBER	Decimal	Last sequence number written to disk. If a sequence uses caching, the number written to disk is the last number placed in the sequence cache. This number is likely to be greater than the last sequence number that was used.

Functions

[Table A-13](#) lists the column name, data type, and description of the Functions Schema Collection.

Table A-13 Functions

Column Name	Data Type	Description
OWNER	String	Owner of the function.
OBJECT_NAME	String	Name of the function.
SUBOBJECT_NAME	String	Name of the subobject (for example, partition).
OBJECT_ID	Decimal	Dictionary object number of the function.
DATA_OBJECT_ID	Decimal	Dictionary object number of the segment that contains the function.
CREATED	DateTime	Timestamp for the creation of the function.

Table A-13 (Cont.) Functions

Column Name	Data Type	Description
LAST_DDL_TIME	DateTime	Timestamp for the last modification of the function resulting from a DDL statement (including grants and revokes).
TIMESTAMP	String	Timestamp for the specification of the function (character data).
STATUS	String	Status of the function (VALID, INVALID, or N/A).
TEMPORARY	String	Whether or not the function is temporary (the current session can see only data that it placed in this object itself).
GENERATED	String	Indicates whether the name of this function is system generated (Y) or not (N).
SECONDARY	String	Whether or not this is a secondary object created by the ODCIIndexCreate method of the Oracle Data Cartridge (Y N).

Procedures

[Table A-14](#) lists the column name, data type, and description of the Procedures Schema Collection.

Table A-14 Procedures

Column Name	Data Type	Description
OWNER	String	Owner of the procedure.
OBJECT_NAME	String	Name of the procedure.
SUBOBJECT_NAME	String	Name of the subobject (for example, partition).
OBJECT_ID	Decimal	Dictionary object number of the procedure.
DATA_OBJECT_ID	Decimal	Dictionary object number of the segment that contains the procedure.
CREATED	DateTime	Timestamp for the creation of the procedure.
LAST_DDL_TIME	Decimal	Timestamp for the last modification of the procedure resulting from a DDL statement (including grants and revokes).
TIMESTAMP	String	Timestamp for the specification of the procedure (character data).
STATUS	String	Status of the procedure (VALID, INVALID, or N/A).
TEMPORARY	String	Whether or not the procedure is temporary (the current session can see only data that it placed in this object itself).
GENERATED	String	Indicates whether the name of this procedure is system generated (Y) or not (N).
SECONDARY	String	Whether or not this is a secondary object created by the ODCIIndexCreate method of the Oracle Data Cartridge (Y N).

ProcedureParameters

[Table A-15](#) lists the column name, data type and description of the ProcedureParameters Schema Collection.

Table A-15 ProcedureParameters

Column Name	Data Type	Description
OWNER	String	Owner of the object.
OBJECT_NAME	String	Name of the procedure or function.
PACKAGE_NAME	String	Name of the package.
OBJECT_ID	Decimal	Object number of the object.
OVERLOAD	String	Indicates the <i>n</i> th overloading ordered by its appearance in the source; otherwise, it is NULL.
SUBPROGRAM_ID	Decimal	Subprogram id for the procedure or function
ARGUMENT_NAME	String	If the argument is a scalar type, then the argument name is the name of the argument. A null argument name is used to denote a function return value.
POSITION	Decimal	If DATA_LEVEL is zero, then this column holds the position of this item in the argument list, or zero for a function return value.
SEQUENCE	Decimal	Defines the sequential order of the argument. Argument sequence starts from 1.
DATA_LEVEL	Decimal	Nesting depth of the argument for composite types.
DATA_TYPE	String	Data type of the argument.
DEFAULT_VALUE	String	Default value for the argument.
DEFAULT_LENGTH	Decimal	Length of the default value for the argument.
IN_OUT	String	Direction of the argument: [IN] [OUT] [IN/OUT].
DATA_LENGTH	Decimal	Length of the column (in bytes).
DATA_PRECISION	Decimal	Length in decimal digits (NUMBER) or binary digits (FLOAT).
DATA_SCALE	Decimal	Digits to the right of the decimal point in a number.
RADIX	Decimal	Argument radix for a number.
CHARACTER_SET_NAME	String	Character set name for the argument.
TYPE_OWNER	String	Owner of the type of the argument.
TYPE_NAME	String	Name of the type of the argument. If the type is a package local type (that is, it is declared in a package specification), then this column displays the name of the package.
TYPE_SUBNAME	String	Displays the name of the type declared in the package identified in the TYPE_NAME column. Relevant only for package local types.
TYPE_LINK	String	Displays the database link that refers to the remote package. Relevant only for package local types when the package identified in the TYPE_NAME column is a remote package.

Table A-15 (Cont.) ProcedureParameters

Column Name	Data Type	Description
PLS_TYPE	String	For numeric arguments, the name of the PL/SQL type of the argument. Otherwise, Null.
CHAR_LENGTH	Decimal	Character limit for string data types.
CHAR_USED	String	Indicates whether the byte limit (B) or character limit (C) is official for the string.

Arguments

[Table A-16](#) lists the column name, data type, and description of the Arguments Schema Collection.

Table A-16 Arguments

Column Name	Data Type	Description
OWNER	String	Owner of the object.
PACKAGE_NAME	String	Name of the package.
OBJECT_NAME	String	Name of the procedure or function.
ARGUMENT_NAME	String	If the argument is a scalar type, then the argument name is the name of the argument. A null argument name is used to denote a function return value.
POSITION	Decimal	If DATA_LEVEL is zero, then this column holds the position of this item in the argument list, or zero for a function return value.
SEQUENCE	Decimal	Defines the sequential order of the argument. Argument sequence starts from 1.
DEFAULT_VALUE	String	Default value for the argument.
DEFAULT_LENGTH	Decimal	Length of the default value for the argument.
IN_OUT	String	Direction of the argument: [IN] [OUT] [IN/OUT].
DATA_LENGTH	Decimal	Length of the column (in bytes).
DATA_PRECISION	Decimal	Length in decimal digits (NUMBER) or binary digits (FLOAT).
DATA_SCALE	Decimal	Digits to the right of the decimal point in a number.
DATA_TYPE	String	Data type of the argument.
CHAR_USED	String	Indicates whether the column uses BYTE length semantics (B) or CHAR length semantics (C).

Packages

[Table A-17](#) lists the column name, data type, and description of the Packages Schema Collection.

Table A-17 Packages

Column Name	Data Type	Description
OWNER	String	Owner of the package.
OBJECT_NAME	String	Name of the package.
SUBOBJECT_NAME	String	Name of the subobject (for example, partition).
OBJECT_ID	Decimal	Dictionary object number of the package.
DATA_OBJECT_ID	Decimal	Dictionary object number of the segment that contains the package.
CREATED	DateTime	Timestamp for the creation of the package.
LAST_DDL_TIME	DateTime	Timestamp for the last modification of the package resulting from a DDL statement (including grants and revokes).
TIMESTAMP	String	Timestamp for the specification of the package (character data).
STATUS	String	Status of the package (VALID, INVALID, or N/A).
TEMPORARY	String	Whether or not the package is temporary (the current session can see only data that it placed in this object itself).
GENERATED	String	Indicates whether the name of this package was system generated (Y) or not (N).
SECONDARY	String	Whether or not this is a secondary object created by the ODCIIndexCreate method of the Oracle Data Cartridge (Y N).

PackageBodies

[Table A-18](#) lists the column name, data type, and description of the PackageBodies Schema Collection.

Table A-18 PackageBodies

Column Name	Data Type	Description
OWNER	String	Owner of the package body.
OBJECT_NAME	String	Name of the package body.
SUBOBJECT_NAME	String	Name of the subobject (for example, partition).
OBJECT_ID	Decimal	Dictionary object number of the package body.
DATA_OBJECT_ID	Decimal	Dictionary object number of the segment that contains the package body.
CREATED	DateTime	Timestamp for the creation of the package body.
LAST_DDL_TIME	DateTime	Timestamp for the last modification of the package body resulting from a DDL statement (including grants and revokes).
TIMESTAMP	String	Timestamp for the specification of the package body (character data).
STATUS	String	Status of the package body (VALID, INVALID, or N/A).

Table A-18 (Cont.) PackageBodies

Column Name	Data Type	Description
TEMPORARY	String	Whether the package body is temporary (the current session can see only data that it placed in this object itself).
GENERATED	String	Indicates whether the name of this package body is system generated (Y) or not (N).
SECONDARY	String	Whether or not this is a secondary object created by the ODCIIndexCreate method of the Oracle Data Cartridge (Y N).

JavaClasses

[Table A-19](#) lists the column name, data type, and description of the JavaClasses Schema Collection.

Table A-19 JavaClasses

Column Name	Data Type	Description
OWNER	String	Owner of the Java class.
NAME	String	Name of the Java class.
MAJOR	Decimal	Major version number of the Java class, as defined in the JVM specification.
MINOR	Decimal	Minor version number of the Java class, as defined in the JVM specification.
KIND	String	Indicates whether the stored object is a Java class (CLASS) or a Java interface (INTERFACE).
ACCESSIBILITY	String	Accessibility of the Java class.
IS_INNER	String	Indicates whether this Java class is an inner class (YES) or not (NO).
IS_ABSTRACT	String	Indicates whether this Java class is an abstract class (YES) or not (NO).
IS_FINAL	String	Indicates whether this Java class is a final class (YES) or not (NO).
IS_DEBUG	String	Indicates whether this Java class contains debug information (YES) or not (NO).
SOURCE	String	Source designation of the Java class.
SUPER	String	Super class of this Java class.
OUTER	String	Outer class of this Java class if this Java class is an inner class.

Indexes

[Table A-20](#) lists the column name, data type, and description of the Indexes Schema Collection.

Table A-20 Indexes

Column Name	Data Type	Description
OWNER	String	Owner of the index.
INDEX_NAME	String	Name of the index.
INDEX_TYPE	String	Type of the index: - NORMAL - BITMAP - FUNCTION-BASED NORMAL - FUNCTION-BASED BITMAP - DOMAIN
TABLE_OWNER	String	Owner of the indexed object.
TABLE_NAME	String	Name of the indexed object.
TABLE_TYPE	String	Type of the indexed object (for example, TABLE or CLUSTER).
UNIQUENESS	String	Indicates whether the index is UNIQUE or NONUNIQUE.
COMPRESSION	String	Indicates whether index compression is enabled (ENABLED) or not (DISABLED).
PREFIX_LENGTH	Decimal	Number of columns in the prefix of the compression key.
TABLESPACE_NAME	String	Name of the tablespace containing the index.
INI_TRANS	Decimal	Initial number of transactions.
MAX_TRANS	Decimal	Maximum number of transactions.
INITIAL_EXTENT	Decimal	Size of the initial extent.
NEXT_EXTENT	Decimal	Size of secondary extents.
MIN_EXTENTS	Decimal	Minimum number of extents allowed in the segment.
MAX_EXTENTS	Decimal	Maximum number of extents allowed in the segment.
PCT_INCREASE	Decimal	Percentage increase in extent size.
PCT_THRESHOLD	Decimal	Threshold percentage of block space allowed per index entry.
INCLUDE_COLUMN	Decimal	Column ID of the last column to be included in index-organized table primary key (non-overflow) index. This column maps to the COLUMN_ID column of the *_TAB_COLUMNS data dictionary views.
FREELISTS	Decimal	Number of process freelists allocated to this segment.
FREELIST_GROUPS	Decimal	Number of freelist groups allocated to this segment.
PCT_FREE	Decimal	Minimum percentage of free space in a block.
LOGGING	String	Logging information.
BLEVEL	Decimal	B*-Tree level: depth of the index from its root block to its leaf blocks. A depth of 0 indicates that the root block and leaf block are the same.
LEAF_BLOCKS	Decimal	Number of leaf blocks in the index.
DISTINCT_KEYS	Decimal	Number of distinct indexed values. For indexes that enforce UNIQUE and PRIMARY KEY constraints, this value is the same as the number of rows in the table (USER_TABLES.NUM_ROWS).

Table A-20 (Cont.) Indexes

Column Name	Data Type	Description
AVG_LEAF_BLOCKS_PER_KEY	Decimal	Average number of leaf blocks in which each distinct value in the index appears, rounded to the nearest integer. For indexes that enforce UNIQUE and PRIMARY KEY constraints, this value is always 1.
AVG_DATA_BLOCKS_PER_KEY	Decimal	Average number of data blocks in the table that are pointed to by a distinct value in the index, rounded to the nearest integer. This statistic is the average number of data blocks that contain rows that contain a given value for the indexed columns.
CLUSTERING_FACTOR	Decimal	Indicates the amount of order of the rows in the table based on the values of the index.
STATUS	String	Indicates whether a nonpartitioned index is VALID or UNUSABLE.
NUM_ROWS	Decimal	Number of rows in the index.
SAMPLE_SIZE	Decimal	Size of the sample used to analyze the index.
LAST_ANALYZED	Date	Date on which this index was most recently analyzed.
DEGREE	String	Number of threads per instance for scanning the index.
INSTANCES	String	Number of instances across which the indexes to be scanned.
PARTITIONED	String	Indicates whether the index is partitioned (YES) or not (NO).
TEMPORARY	String	Indicates whether or not the index is on a temporary table.
GENERATED	String	Indicates whether the name of the index is system generated (Y) or not (N).
SECONDARY	String	Indicates whether the index is a secondary object created by the ODCIIndexCreate method of the Oracle Data Cartridge (Y) or not (N).
BUFFER_POOL	String	Name of the default buffer pool to be used for the index blocks.
USER_STATS	String	Indicates whether statistics were entered directly by the user (YES) or not (NO).
DURATION	String	Indicates the duration of a temporary table.
PCT_DIRECT_ACCESS	Decimal	For a secondary index on an index-organized table, the percentage of rows with VALID guess.
ITYP_OWNER	String	For a domain index, the owner of the index type.
ITYP_NAME	String	For a domain index, the name of the index type.
PARAMETERS	String	For a domain index, the parameter string.
GLOBAL_STATS	String	For partitioned indexes, indicates whether statistics are collected by analyzing the index as a whole (YES) or estimated from statistics on underlying index partitions and subpartitions (NO).

Table A-20 (Cont.) Indexes

Column Name	Data Type	Description
DOMIDX_STATUS	String	Status of the domain index: • NULL - Index is not a domain index. • VALID - Index is a valid domain index. • IDXTYP_INVLD - Indextype of the domain index is invalid.
DOMIDX_OPSTATUS	String	Status of the operation on the domain index: • NULL - Index is not a domain index. • VALID - Operation performed without errors. • FAILED - Operation failed with an error.
FUNCIDX_STATUS	String	Status of a function-based index: • NULL - Index is not a function-based index. • ENABLED - Function-based index is enabled. • DISABLED - Function-based index is disabled.
JOIN_INDEX	String	Indicates whether the index is a join index (YES) or not (NO).
IOT_REDUNDANT_PKEY_ELIM	String	Indicates whether redundant primary key columns are eliminated from secondary indexes on index-organized tables (YES) or not (NO).
DROPPED	String	Indicates whether the index has been dropped and is in the recycle bin (YES) or not (NO); null for partitioned tables.

IndexColumns

[Table A-21](#) lists the column name, data type, and description of the IndexColumns Schema Collection.

Table A-21 IndexColumns

Column Name	Data Type	Description
INDEX_OWNER	String	Owner of the index.
INDEX_NAME	String	Name of the index.
TABLE_OWNER	String	Owner of the table or cluster.
TABLE_NAME	String	Name of the table or cluster.
COLUMN_NAME	String	Column name or attribute of object type column.
COLUMN_POSITION	Decimal	Position of column or attribute within the index.
COLUMN_LENGTH	Decimal	Indexed length of the column.
DESCEND	String	Whether the column is sorted in descending order (Y/N).
CHAR_LENGTH	Decimal	Maximum codepoint length of the column. (Oracle9i or later)

PrimaryKeys

[Table A-22](#) lists the column name, data type, and description of the PrimaryKeys Schema Collection.

Table A-22 PrimaryKeys

Column Name	Data Type	Description
OWNER	String	Owner of the constraint definition.
CONSTRAINT_NAME	String	Name of the constraint definition.
TABLE_NAME	String	Name associated with the table (or view) with constraint definition.
SEARCH_CONDITION	String	Text of search condition for a check constraint.
R_OWNER	String	Owner of table referred to in a referential constraint.
R_CONSTRAINT_NAME	String	Name of the unique constraint definition for referenced table.
DELETE_RULE	String	Delete rule for a referential constraint (CASCADE or NO ACTION).
STATUS	String	Enforcement status of constraint (ENABLED or DISABLED).
DEFERRABLE	String	Whether or not the constraint is deferrable.
VALIDATED	String	Whether all data obeys the constraint (VALIDATED or NOT VALIDATED).
GENERATED	String	Whether the name of the constraint is user or system generated.
BAD	String	Indicates that this constraint specifies a century in an ambiguous manner. (Yes No) To avoid errors resulting from this ambiguity, rewrite the constraint using the TO_DATE function with a four-digit year.
RELY	String	Whether an enabled constraint is enforced or unenforced.
LAST_CHANGE	DateTime	When the constraint was last enabled or disabled.
INDEX_OWNER	String	Name of the user owning the index. (Oracle9i or later)
INDEX_NAME	String	Name of the index (only shown for unique and primary-key constraints). (Oracle9i or later)

ForeignKeys

[Table A-23](#) lists the column name, data type, and description of the ForeignKeys Schema Collection.

Table A-23 ForeignKeys

Column Name	Data Type	Description
PRIMARY_KEY_CONSTRAINT_NAME	String	Name of the constraint definition.
PRIMARY_KEY_OWNER	String	Owner of the constraint definition.
PRIMARY_KEY_TABLE_NAME	String	Name associated with the table (or view) with constraint definition.

Table A-23 (Cont.) ForeignKeys

Column Name	Data Type	Description
FOREIGN_KEY_OWNER	String	Owner of the constraint definition.
FOREIGN_KEY_CONSTR INT_NAME	String	Name of the constraint definition.
FOREIGN_KEY_TABLE_N AME	String	Name associated with the table (or view) with constraint definition.
SEARCH_CONDITION	String	Text of search condition for a check constraint
R_OWNER	String	Owner of table referred to, in a referential constraint.
R_CONSTRAINT_NAME	String	Name of the unique constraint definition for referenced table.
DELETE_RULE	String	Delete rule for a referential constraint (CASCADE or NO ACTION).
STATUS	String	Enforcement status of constraint (ENABLED or DISABLED).
VALIDATED	String	Whether or not all data obeys the constraint (VALIDATED or NOT VALIDATED).
GENERATED	String	Whether the name of the constraint is user or system generated.
RELY	String	Whether an enabled constraint is enforced or unenforced.
LAST_CHANGE	DateTime	When the constraint was last enabled or disabled.
INDEX_OWNER	String	Name of the user owning the index. (Oracle9i or later)
INDEX_NAME	String	Name of the index. (Oracle9i or later)

ForeignKeyColumns

[Table A-24](#) lists the column name, data type, and description of the ForeignKeyColumns Schema Collection.

Table A-24 ForeignKeyColumns

Column Name	Data Type	Description
OWNER	String	Owner of the constraint definition.
CONSTRAINT_NAME	String	Name of the constraint definition.
TABLE_NAME	String	Name of the table with constraint definition.
COLUMN_NAME	String	Name of the column or attribute of the object type column specified in the constraint definition.
POSITION	String	Original position of column or attribute in the definition of the object.

UniqueKeys

[Table A-25](#) lists the column name, data type, and description of the UniqueKeys Schema Collection.

Table A-25 UniqueKeys

Column Name	Data Type	Description
OWNER	String	Owner of the constraint definition.
CONSTRAINT_NAME	String	Name of the constraint definition.
TABLE_NAME	String	Name associated with the table (or view) with constraint definition.
SEARCH_CONDITION	String	Text of search condition for a check constraint.
R_OWNER	String	Owner of table referred to in a referential constraint.
R_CONSTRAINT_NAME	String	Name of the unique constraint definition for referenced table.
DELETE_RULE	String	Delete rule for a referential constraint (CASCADE or NO ACTION).
STATUS	String	Enforcement status of constraint (ENABLED or DISABLED).
DEFERRABLE	String	Whether or not the constraint is deferrable.
VALIDATED	String	Whether all data obeys the constraint (VALIDATED or NOT VALIDATED).
GENERATED	String	Whether the name of the constraint is user or system generated.
BAD	String	Indicates that this constraint specifies a century in an ambiguous manner. (Yes No) To avoid errors resulting from this ambiguity, rewrite the constraint using the TO_DATE function with a four-digit year.
RELY	String	Whether an enabled constraint is enforced or not.
LAST_CHANGE	String	When the constraint was last enabled or disabled.
INDEX_OWNER	String	Name of the user owning the index. (Oracle9i or later)
INDEX_NAME	String	Name of the index (only shown for unique and primary-key constraints). (Oracle9i or later)

B

Mapping LINQ Canonical Functions and Oracle Functions

This appendix lists the Entity Framework canonical functions and the corresponding ODP.NET provider functions to which they map.

Aggregate Canonical Functions

Table B-1 Mapping of Aggregate Canonical Functions and Oracle Functions

Canonical Function	Oracle Function
Avg (<i>expression</i>)	AVG(<i>expression</i>)
BigCount (<i>expression</i>)	COUNT(<i>expression</i>)
Count (<i>expression</i>)	COUNT(<i>expression</i>)
Max (<i>expression</i>)	MAX(<i>expression</i>)
Min (<i>expression</i>)	MIN(<i>expression</i>)
StDev (<i>expression</i>)	STDDEV(<i>expression</i>)
StDevP(<i>expression</i>)	STDEVP(<i>expression</i>)
Sum (<i>expression</i>)	SUM (<i>expression</i>)
Var(<i>expression</i>)	VAR(<i>expression</i>)
VarP(<i>expression</i>)	VARP(<i>expression</i>)

Math Canonical Functions

Table B-2 Mapping of Math Canonical Functions and Oracle Functions

Canonical Function	Oracle Function
Abs (<i>value</i>)	ABS (<i>value</i>)
Ceiling (<i>value</i>)	CEIL(<i>value</i>)
Floor (<i>value</i>)	FLOOR(<i>value</i>)
Power(<i>value</i> , <i>exponent</i>)	POWER(<i>value</i> , <i>exponent</i>)
Round (<i>value</i>)	ROUND(<i>value</i>)
Round (<i>value</i> , <i>digits</i>)	ROUND(<i>value</i> , <i>digits</i>)
Truncate(<i>value</i> , <i>digits</i>)	TRUNC(<i>value</i> , <i>digits</i>)

String Canonical Functions

Table B-3 Mapping of String Canonical Functions and Oracle Functions

Canonical Function	Oracle Function
Concat (<i>string1</i> , <i>string2</i>)	CONCAT(<i>string1</i> , <i>string2</i>) or ((<i>string1</i>) (<i>string2</i>))
Contains(<i>string</i> , <i>target</i>)	INSTR(<i>string</i> , <i>target</i>)
EndsWith(<i>string</i> , <i>target</i>)	INSTR(REVERSE(<i>string</i>), REVERSE(<i>target</i>))
Comparison operators (<i><</i> , <i><=</i> , <i>></i> , <i>>=</i> , <i><></i> , <i>!=</i>)	Comparison operators (<i><</i> , <i><=</i> , <i>></i> , <i>>=</i> , <i><></i> , <i>!=</i>)
IndexOf(<i>target</i> , <i>string</i>)	INSTR(<i>string2</i> , <i>target</i>)
Left (<i>string1</i> , <i>length</i>)	SUBSTR(<i>string1</i> , <i>length</i>)
Length (<i>string</i>)	LENGTH(<i>string</i>)
LTrim(<i>string</i>)	LTRIM(<i>string</i>)
Replace (<i>string1</i> , <i>string2</i> , <i>string3</i>)	REPLACE(<i>string1</i> , <i>string2</i> , <i>string3</i>)
Reverse (<i>string</i>)	REVERSE(<i>string</i>)
Right (<i>string</i> , <i>length</i>)	(CASE WHEN LENGTH(<i>string</i>) >= (<i>length</i>) THEN SUBSTR(<i>string</i>) ,-(<i>length</i>), <i>length</i>) ELSE <i>string</i> END)
RTrim(<i>string</i>)	RTRIM(<i>string</i>)
Substring (<i>string</i> , <i>start</i> , <i>length</i>)	SUBSTR((<i>string</i> , <i>start</i> , <i>length</i>)
StartsWith(<i>string</i> , <i>target</i>)	INSTR(<i>string</i> , <i>target</i>)
ToLower (<i>string</i>)	LOWER(<i>string</i>)
ToUpper(<i>string</i>)	UPPER
Trim (<i>string</i>)	LTRIM(RTRIM(<i>string</i>))

Date And Time Canonical Functions

Table B-4 Mapping of Date And Time Canonical Functions and Oracle Functions

Canonical Function	Oracle Function
AddNanoseconds(<i>expression</i> , <i>number</i>)	(<i>expression</i>) + INTERVAL
AddMicroseconds(<i>expression</i> , <i>number</i>)	(<i>expression</i>) + INTERVAL
AddMilliseconds(<i>expression</i> , <i>number</i>)	(<i>expression</i>) + INTERVAL
AddSeconds(<i>expression</i> , <i>number</i>)	(<i>expression</i>) + INTERVAL
AddMinutes(<i>expression</i> , <i>number</i>)	(<i>expression</i>) + INTERVAL
AddHours(<i>expression</i> , <i>number</i>)	(<i>expression</i>) + INTERVAL
AddDays(<i>expression</i> , <i>number</i>)	(<i>expression</i>) + INTERVAL

Table B-4 (Cont.) Mapping of Date And Time Canonical Functions and Oracle Functions

Canonical Function	Oracle Function
AddMonths(<i>expression</i> , <i>number</i>)	(<i>expression</i>) + INTERVAL
AddYears(<i>expression</i> , <i>number</i>)	(<i>expression</i>) + INTERVAL
CreateDateTime(<i>year</i> , <i>month</i> , <i>day</i> , <i>hour</i> , <i>minute</i> , <i>second</i>)	TO_TIMESTAMP
CreateDateTimeOffset(<i>year</i> , <i>month</i> , <i>day</i> , <i>hour</i> , <i>minute</i> , <i>second</i> , <i>tzoffset</i>)	TO_TIMESTAMP_TZ
CreateTime(<i>hour</i> , <i>minute</i> , <i>second</i>)	Time literals are not supported in Oracle
CurrentDateTime()	LOCALTIMESTAMP
CurrentDateTimeOffset()	SYSTIMESTAMP
CurrentUtcDateTime()	SYS_EXTRACT_UTC (LOCALTIMESTAMP)
Day(<i>expression</i>)	EXTRACT(DAY FROM <i>expression</i>)
DayOfYear(<i>expression</i>)	TO_NUMBER(TO_CHAR(CAST(<i>expression</i> AS TIMESTAMP), 'DDD'))
DiffNanoseconds(<i>startExpression</i> , <i>endExpression</i>)	EXTRACT and arithmetic operations
DiffMilliseconds(<i>startExpression</i> , <i>endExpression</i>)	EXTRACT and arithmetic operations
DiffMicroseconds(<i>startExpression</i> , <i>endExpression</i>)	EXTRACT and arithmetic operations
DiffSeconds(<i>startExpression</i> , <i>endExpression</i>)	EXTRACT and arithmetic operations
DiffMinutes(<i>startExpression</i> , <i>endExpression</i>)	EXTRACT and arithmetic operations
DiffHours(<i>startExpression</i> , <i>endExpression</i>)	EXTRACT and arithmetic operations
DiffDays(<i>startExpression</i> , <i>endExpression</i>)	EXTRACT and arithmetic operations
DiffMonths(<i>startExpression</i> , <i>endExpression</i>)	EXTRACT and arithmetic operations
DiffYears(<i>startExpression</i> , <i>endExpression</i>)	EXTRACT and arithmetic operations
)	
Comparison operators (<i><</i> , <i><=</i> , <i>></i> , <i>>=</i> , <i><></i> , <i>!=</i>)	<i><</i> , <i><=</i> , <i>></i> , <i>>=</i> , <i><></i> , <i>!=</i> operators
GetTotalOffsetMinutes (<i>datetimeoffset</i>)	(EXTRACT(TIMEZONE_HOUR FROM (<i>expression</i>))) * 60 + EXTRACT (TIMEZONE_MINUTE FROM(<i>expression</i>)) (Require multiple operations.)
Hour (<i>expression</i>)	EXTRACT(HOUR FROM <i>expression</i>)

Table B-4 (Cont.) Mapping of Date And Time Canonical Functions and Oracle Functions

Canonical Function	Oracle Function
Millisecond(<i>expression</i>)	NVL(TO_NUMBER(SUBSTR(TO_CHAR(CAST(<i>expression</i> AS TIMESTAMP), 'DD- MON-RR HH24:MI:SSXFF'), 20, 3)), 0)
Minute(<i>expression</i>)	EXTRACT(MINUTE FROM <i>expression</i>)
Month (<i>expression</i>)	EXTRACT(MONTH FROM <i>expression</i>)
Second(<i>expression</i>)	EXTRACT (SECOND FROM <i>expression</i>)
TruncateDate(<i>expression</i>)	TRUNC(<i>expression</i>)
Year(<i>expression</i>)	EXTRACT(YEAR FROM <i>expression</i>)

Bitwise Canonical Functions**Table B-5 Mapping of Bitwise Canonical Functions and Oracle Functions**

Canonical Function	Oracle Function
BitWiseAnd (<i>value1</i> , <i>value2</i>)	BITAND(<i>value1</i> , <i>value2</i>)
BitWiseNot (<i>value</i>)	(0 - <i>value</i>) - 1
BitWiseOr (<i>value1</i> , <i>value2</i>)	Value1 - BITAND(<i>value1</i> , <i>value2</i>) + <i>value2</i>
BitWiseXor (<i>value1</i> , <i>value2</i>)	Value1 - 2 * BITAND(<i>value1</i> , <i>value2</i>) + <i>value2</i>

Other Canonical Functions**Table B-6 Mapping of Other Canonical Functions and Oracle Functions**

Canonical Function	Oracle Function
NewGuid()	SYS_GUID

Glossary

assembly

Assembly is Microsoft's term for the module that is created when a DLL or .EXE is compiled by a .NET compiler.

BFILES

External binary files that exist outside the database tablespaces residing in the operating system. BFILES are referenced from the database semantics, and are also known as external LOBs.

Binary Large Object (BLOB)

A large object data type whose content consists of binary data. Additionally, this data is considered raw as its structure is not recognized by the database.

Character Large Object (CLOB)

The LOB data type whose value is composed of character data corresponding to the database character set. A CLOB may be indexed and searched by the Oracle Text search engine.

data provider

As the term is used with Oracle Data Provider for .NET, a data provider is the connected component in the ADO.NET model and transfers data between a data source and the DataSet.

DataSet

A DataSet is an in-memory copy of database data. The DataSet exists in memory without an active connection to the database.

dirty writes

Dirty writes means writing uncommitted or dirty data.

DDL

DDL refers to data definition language, which includes statements defining or changing data structure.

DOM

Document Object Model (DOM) is an application program interface (API) for HTML and XML documents. It defines the logical structure of documents and the way that a document is accessed and manipulated.

Extensible Stylesheet Language Transformation (XSLT)

The XSL W3C standard specification that defines a transformation language to convert one XML document into another.

flush

Flush or flushing refers to recording changes (that is, sending modified data) to the database.

Global Assembly Cache (GAC)

A cache for .NET assemblies.

goodness

The degree of load in the Oracle database. The lighter load is better and vice versa.

instantiate

A term used in object-based languages such as C# to refer to the creation of an object of a specific class.

invalidation message

The content of a change notification which indicates that the cache is now invalid

Large Object (LOB)

The class of SQL data type that is further divided into internal LOBs and external LOBs. Internal LOBs include BLOBs, CLOBs, and NLOBs while external LOBs include BFILEs.

Microsoft .NET Framework Class Library

The Microsoft .NET Framework Class Library provides the classes for the .NET framework model.

namespace

- .NET:

A namespace is naming device for grouping related types. More than one namespace can be contained in an assembly.

- XML Documents:

A namespace describes a set of related element names or attributes within an XML document.

National Character Large Object (NCLOB)

The LOB data type whose value is composed of character data corresponding to the database national character set.

Oracle Net Services

The Oracle client/server communication software that offers transparent operation to Oracle tools or databases over any type of network protocol and operating system.

OracleDataReader

An OracleDataReader is a read-only, forward-only result set.

Oracle XML DB

Oracle XML DB is the name for a distinct group of technologies related to high-performance XML storage and retrieval that are available within the Oracle database. Oracle XML DB is not a separate server.

Oracle XML DB is based on the W3C XML data model.

PL/SQL

The Oracle procedural language extension to SQL.

primary key

The column or set of columns included in the definition of a table's PRIMARY KEY constraint.

reference semantics

Reference semantics indicates that assignment is to a reference (an address such as a pointer) rather than to a value. See [value semantics](#).

REF

A data type that encapsulates references to row objects of a specified object type.

result set

The output of a SQL query, consisting of one or more rows of data.

Safe Type Mapping

Safe Type Mapping allows the `OracleDataAdapter` to populate a `DataSet` with .NET type representations of Oracle data without any data or precision loss.

savepoint

A point in the workspace to which operations can be rolled back.

stored procedure

A stored procedure is a PL/SQL block that Oracle stores in the database and can be executed from an application.

Transparent Application Failover (TAF)

Transparent Application Failover is a runtime failover for high-availability environments. It enables client applications to automatically reconnect to the database if the connection fails. This reconnect happens automatically from within the Oracle Call Interface (OCI) library.

Unicode

Unicode is a universal encoded character set that enables information from any language to be stored using a single character set.

URL

URL (Universal Resource Locator).

value semantics

Value semantics indicates that assignment copies the value, not the reference or address (such as a pointer). See [reference semantics](#).

XPath

XML Path Language (XPath), based on a W3C recommendation, is a language for addressing parts of an XML document. It is designed to be used by both XSLT and XPointer. It can be used as a searching or query language as well as in hypertext linking.

Index

Symbols

.NET custom types, [178](#)
.NET Framework datatype, [120](#)
.NET languages, [1](#), [2](#)
.NET products and documentation, [1](#)
.NET Stream class, [151](#)
.NET type accessors, [126](#)
.NET Types
 inference, [104](#)

A

abstract data types, [178](#)
ADO, [3](#)
ADO.NET, [3](#), [191](#)
ADO.NET 1.x, [1](#)
ADO.NET 2.0, [6](#)
 base or DbCommon classes, [2](#)
ADO.NET 2.x, [1](#)
ADTs, [178](#)
app.config file, [30](#)
AppDomain, [59](#)
application config, [11](#)
array bind
 OracleParameter, [112](#)
array bind operations, [111](#)
 ArrayBindCount, [27](#)
 ArrayBindIndex, [561](#)
 ArrayBindSize, [589](#), [591](#), [622](#)
 ArrayBindStatus, [623](#)
 error handling, [112](#)
array binding, [111](#)
ArrayBindCount property, [27](#)
ArrayBindIndex property, [561](#)
ArrayBindSize property, [108](#), [112](#), [622](#)
ArrayBindStatus property, [108](#), [112](#), [623](#)
ASP.NET, [2](#)
assembly, [4](#)
 ODP.NET, [4](#)
AzureTokenAuthentication
 constructors, [3](#)
AzureTokenAuthentication Class
 class description, [1](#)
 members, [2](#)
 properties, [4](#)

B

batch processing, [114](#)
BatchUpdate
 Microsoft Hotfix, [114](#)
behavior of ExecuteScalar method for REF
 CURSOR, [140](#)
BFILE, [150](#)
binding, [102](#)
 PL/SQL Associative Array, [108](#)
BLOB, [150](#)
bulk copy constraints, [196](#)
bulk copy feature, [196](#)
 restrictions, [196](#)

C

C#, Visual Basic .NET, C++ .NET, [1](#)
case-sensitivity
 column name mapping, [175](#)
change notification
 ODP.NET support, [212](#)
change notification, Continuous Query
 Notification, [1](#)
characters with special meaning
 in column data, [174](#)
 in table or view, [175](#)
characters with special meaning in XML, [169](#)
client applications, [1](#)
client globalization settings, [223](#), [226](#)
client identifier, [72](#)
CLOB, [150](#)
close calls, [78](#)
collection types, [178](#)
CollectionType property, [108](#)
column data
 special characters in, [174](#)
CommandBehavior.SequentialAccess, [131](#)
commit transactions
 changes to XML data, [177](#)
CommittableTransaction, [95](#)
Committed property, [589](#)
configuration settings
 UDTs, [192](#)
connect descriptor, [30](#)
connection dependency, [167](#)

- connection pool
 - performance counters, [56](#)
- connection pooling, [53](#)
 - example, [53](#)
 - management, [55](#)
- ConnectionString attributes, [53](#)
 - Connection Lifetime, [27](#), [53](#), [54](#)
 - Connection Timeout, [27](#), [53](#), [54](#)
 - Data Source, [27](#)
 - DBA Privilege, [27](#)
 - Decr Pool Size, [27](#), [53](#), [54](#)
 - Enlist, [27](#)
 - HA Events, [27](#), [53](#)
 - Incr Pool Size, [27](#), [53](#), [54](#)
 - Load Balancing, [27](#), [53](#)
 - Max Pool Size, [27](#), [53](#), [54](#)
 - Metadata Pooling, [28](#)
 - Min Pool Size, [27](#), [53](#), [54](#)
 - Password, [27](#)
 - Persist Security Info, [27](#)
 - Pooling, [27](#), [53](#), [54](#)
 - Proxy Password, [27](#), [71](#)
 - Proxy User Id, [27](#), [71](#)
 - Statement Cache Purge, [28](#)
 - Statement Cache Size, [28](#)
 - User Id, [27](#)
 - Validate Connection, [27](#), [53](#), [54](#)
- ConnectionString property, [53](#), [54](#), [208](#)
- Constraints property, [221](#)
 - configuring, [222](#)
- continuous query notification, [77](#), [211](#)
 - ODP.NET support, [212](#)
- Continuous Query Notification, [209](#)
 - best practices, [216](#)
 - performance considerations, [216](#)
- controlling query reexecution, [220](#)
- custom classes, [179](#)
- custom type factories, [179](#)
- custom type factory, [182](#)
- custom type implementations
 - optional, [180](#)
- custom type mapping, [182](#)
- custom type mappings
 - specifying, [181](#)
 - specifying with custom type factories, [181](#)
 - specifying with XML, [181](#)
 - using, [183](#)
- custom types
 - converting to Oracle UDTs, [184](#)
 - requirements, [179](#)
- custom UDT classes, [179](#)

D

- Data Guard, [74](#)
- data loss, [216](#)

- data manipulation
 - using XML, [173](#)
- data source attribute, [30](#)
- data source enumerators, [34](#)
- database
 - changes to, [173](#)
- database notification
 - port to listen, [77](#)
- DataSet, [152](#)
 - populating, [139](#)
 - populating from a REF CURSOR, [139](#)
 - populating with generic and custom objects, [191](#)
 - updating, [140](#)
 - updating to database, [221](#)
- DataTable, [222](#)
- Datatable properties, [221](#)
- DbProviderFactories class, [6](#), [7](#), [2](#)
- DbType
 - inference, [104](#)
- debug tracing, [237](#)
- default mapping
 - improving, [177](#)
- dependent unmanaged DLL mismatch, [7](#)
- direct path load, [196](#)
- dispose calls, [78](#)
- distributed transactions, [95](#)
- documentation
 - .NET, [1](#)
- Dynamic Enlistment, [72](#)
- dynamic help, [1](#), [6](#), [7](#)

E

- Easy Connect naming method, [31](#)
- EDM type facets, [6](#)
- EDM types, [2](#)
 - and Oracle data types, [2](#)
- end-to-end tracing, [72](#)
- EnlistDistributedTransaction method, [72](#)
- Entity Framework, [1](#)
- enumeration type
 - OracleDbType, [103](#)
- error handling, [112](#)
- example
 - connection pooling, [53](#)
- examples
 - documentation
 - readme file, [6](#), [9](#)
- ExecuteNonQuery method, [140](#)
- ExecuteScalar method, [140](#)
- EZCONNECT, [31](#)

F

FailoverEvent Enumeration
 description, [10](#)
 FailoverReturnCode Enumeration
 description, [12](#)
 FailoverType Enumeration
 description, [12](#)
 Fast Application Notification (FAN), [75](#)
 Fast Connection Failover (FCF), [77](#)
 FCF, [78](#)
 features, [1](#)
 FetchSize property
 fine-tuning, [137](#)
 setting at design time, [137](#)
 setting at run time, [138](#)
 using, [136](#)
 file locations, [6, 9](#)

G

garbage collection, [78](#)
 GDS, [74](#)
 geographic data, [178](#)
 Global Assembly Cache (GAC), [6, 7](#)
 Global Data Services, [76](#)
 Global Data Services (GDS), [74](#)
 globalization settings, [223](#)
 client, [223](#)
 session, [224](#)
 thread-based, [224](#)
 globalization support, [223](#)
 globalization-sensitive operations, [226](#)
 GoldenGate, [74](#)
 GUI access to ODP.NET, [2](#)

H

HA events, [77](#)
 HA Events, [27](#)
 handling date and time format
 manipulating data in XML, [174](#)
 retrieving queries in XML, [170](#)
 high availability, [74](#)

I

implicit REF CURSOR, [142](#)
 bind and metadata, [143](#)
 bindinfo, [143](#)
 configuration, [146](#)
 examples, [146](#)
 usage, [149](#)
 improving default mapping, [177](#)
 inference of DbType and OracleDbType from
 Value, [106](#)

inference of DbType from OracleDbType, [104](#)
 inference of OracleDbType from DbType, [105](#)
 inference of types, [104](#)
 InitialLOBFetchSize property, [132](#)
 InitialLONGFetchSize property, [131](#)
 input binding
 XMLType column, [168](#)
 insert triggers, [198](#)
 installation, [6, 7](#)
 Oracle Data Provider for .NET, [6](#)
 Oracle Data Provider for .NET, Manager
 Provider, [7](#)
 XCopy class, [6, 7](#)
 integrated help, [6, 7](#)
 interference in OracleParameter class, [104](#)
 introduction, overview, [3](#)
 INullable Interface
 interface description, [463](#)
 interface members, [464](#)
 interface properties, [464](#)
 invalidation message, [211](#)
 ensuring persistency of, [212](#)
 IOracleArrayTypeFactory Interface
 interface description, [25](#)
 interface members, [25](#)
 interface methods, [26](#)
 IOracleCustomType Interface
 interface description, [18](#)
 interface members, [18](#)
 interface methods, [19](#)
 IOracleCustomTypeFactory Interface
 interface description, [23](#)
 interface members, [23](#)
 interface methods, [24](#)

L

large binary datatypes, [151](#)
 large character datatypes, [151](#)
 LINQ to Entities, [1](#)
 load balancing, [77](#)
 Load Balancing, [27](#)
 LOBs
 temporary, [153](#)
 updating, [152, 153](#)
 LOBs updating, [152](#)
 local transactions, [95](#)
 location data, [178](#)
 logical transaction ID (LTXID), [80](#)
 LONG and LONG RAW datatypes, [151](#)
 LTXID, [80](#)

M

machine.config, [11](#)
 machine.config file, [6, 7](#)

- metadata, [222](#)
- method invocation
 - UDT, [192](#)
- Microsoft .NET Framework, [1](#)
- Microsoft .NET Framework Class Library, [3](#)
- Microsoft Hotfix
 - BatchUpdate, [114](#)
- multiple notification requests, [212](#)
- multiple tables
 - changes to, [177](#)

N

- namespace
 - Oracle.DataAccess.Types, [15](#)
- native XML support, [165](#)
- NCLOB, [150](#)
- nested table types, [178](#)
- notification framework, [211](#)
- notification information
 - retrieving, [212](#)
- notification process
 - flow, [213](#)
- notification registration, [212](#)
 - requirements of, [213](#)
- NULL values
 - retrieving from column, [173](#)
- number of rows fetched in round-trip
 - controlling, [136](#)

O

- object data type support, [178](#)
- object types, [178](#)
- object-relational data, [173](#)
 - saving changes from XML data, [177](#)
- obtaining a REF CURSOR, [139](#)
- obtaining an OracleRefCursor, [138](#)
- obtaining data from an OracleDataReader, [126](#)
- obtaining LOB data
 - InitialLOBFetchSize property, [132](#)
- obtaining LONG and LONG RAW Data, [131](#)
- OCI
 - statement caching, [114](#)
- OciTokenAuthentication Class
 - class description, [1](#)
 - members, [2](#)
 - properties, [3](#)
- ODP.NET
 - installing, [6](#)
- ODP.NET Configuration, [11](#)
- ODP.NET LOB classes, [150](#)
- ODP.NET Type accessors, [130](#)
- ODP.NET Type classes, [120](#)
- ODP.NET Type exceptions, [1](#)
- ODP.NET Type structures, [120](#), [1](#)

- ODP.NET Types, [120](#)
 - overview, [120](#)
- ODP.NET XML Support, [165](#)
- OnChangedEventArgs Class
 - instance properties, [31](#)
 - members, [29](#)
 - static fields, [30](#)
 - static methods, [30](#)
- OnChangeEventHandler Delegate
 - description, [36](#)
- operating system authentication, [50](#)
- Oracle Call Interface
 - statement caching, [114](#)
- Oracle Data Provider for .NET
 - installing, [6](#)
 - system requirements, [1](#)
- Oracle Data Provider for .NET assembly, [4](#)
- Oracle Data Provider for .NET, Managed Driver
 - installing, [7](#)
- Oracle data types, [2](#)
 - and EDM types, [2](#)
 - mapping and customizing, [13](#)
- Oracle Developer Tools for Visual Studio, [2](#)
- Oracle Label Security, [72](#)
- Oracle native types, [120](#)
 - supported by ODP.NET, [126](#)
- Oracle Providers for ASP.NET, [2](#)
- Oracle RAC, [74](#)
- Oracle Real Application Clusters (Oracle RAC), [74](#)
- Oracle UDT attribute mappings, [185](#)
- Oracle Universal Installer (OUI), [6](#)
- Oracle user-defined types, [178](#)
- Oracle User-Defined Types (UDTs), [178](#)
- Oracle Virtual Private Database (VPD), [72](#)
- Oracle XML DB, [166](#)
- Oracle.DataAccess.Client namespace, [4](#)
- Oracle.DataAccess.dll, [4](#)
- Oracle.DataAccess.Types namespace, [4](#), [15](#)
- OracleAccessToken
 - constructors, [4](#)
- OracleAccessToken Class
 - class description, [2](#)
 - members, [3](#)
- OracleAllowedLogonVersionClient Enumeration, [765](#)
- OracleAQAgent
 - constructors, [3](#)
- OracleAQAgent Class
 - constructors, [3](#)
 - description, [1](#)
 - members, [2](#)
 - properties, [5](#), [9](#)
- OracleAQDequeueMode Enumeration
 - description, [85](#)

- OracleAQDequeueOptions Class
 - constructor, [8](#)
 - description, [6](#)
 - members, [7](#)
 - properties, [9](#)
 - public methods, [15](#)
- OracleAQEnqueueOptions Class
 - constructor, [17](#)
 - description, [16](#)
 - members, [17](#)
 - properties, [18](#)
 - public methods, [19](#)
- OracleAQMessage Class
 - constructors, [23](#)
 - description, [20](#)
 - members, [22](#)
 - properties, [24](#)
- OracleAQMessageAvailableEventArgs Class
 - description, [33](#)
 - members, [34](#)
 - properties, [36](#)
- OracleAQMessageAvailableEventHandler
 - Delegate
 - description, [45](#)
- OracleAQMessageDeliveryMode Enumeration
 - description, [86](#)
- OracleAQMessageState Enumeration
 - description, [88](#)
- OracleAQMessageType Enumeration
 - description, [88](#)
- OracleAQNavigationMode Enumeration
 - description, [89](#)
- OracleAQNotificationGroupingType Enumeration
 - description, [91](#)
- OracleAQNotificationType Enumeration
 - description, [91](#)
- OracleAQQueue Class
 - constructors, [55](#)
 - description, [46](#)
 - events, [81](#)
 - members, [47](#)
 - properties, [49](#)
 - public methods, [62](#)
 - static methods, [55](#)
- OracleAQVisibilityMode Enumeration
 - description, [92](#)
- OracleArrayMappingAttribute Class
 - constructors, [15](#)
 - description, [13](#)
 - members, [14](#)
 - methods, [17](#)
 - properties, [17](#)
 - static methods, [16](#)
- OracleBFile Class
 - class description, [1](#)
 - constructors, [6](#)
- OracleBFile Class (*continued*)
 - instance methods, [17](#)
 - instance properties, [9](#)
 - members, [3](#)
 - static fields, [8](#)
 - static methods, [9](#)
- OracleBinary Structure
 - constructor, [5](#)
 - description, [1](#)
 - instance methods, [23](#)
 - members, [3](#)
 - properties, [20](#)
 - static fields, [5](#)
 - static methods, [6](#)
 - static operators, [13](#)
 - static type conversion operators, [18](#)
- OracleBlob Class
 - class description, [53](#)
 - constructors, [58](#)
 - instance methods, [68](#)
 - instance properties, [61](#)
 - members, [55](#)
 - static fields, [59](#)
 - static methods, [61](#)
- OracleBulkCopy Class
 - class description, [1](#)
 - constructors, [4](#)
 - events, [20](#)
 - members, [2](#)
 - properties, [7](#)
 - public methods, [14](#)
- OracleBulkCopyColumnMapping Class
 - class description, [21](#)
 - constructors, [23](#)
 - members, [22](#)
 - properties, [26, 27](#)
- OracleBulkCopyColumnMappingCollection Class
 - class description, [30](#)
 - members, [31](#)
 - properties, [32](#)
 - public methods, [33](#)
- OracleBulkCopyOptions Enumeration
 - description, [43](#)
- OracleClientFactory, [6, 7](#)
- OracleClientFactory class
 - instantiating, [2](#)
- OracleClientFactory Class
 - class description, [6](#)
 - class members, [7](#)
 - public methods, [11](#)
 - public properties, [10](#)
- OracleClob Class
 - class description, [103](#)
 - constructors, [108](#)
 - instance methods, [119](#)
 - instance properties, [111](#)

- OracleClob Class (*continued*)
 - members, [105](#)
 - static fields, [110](#)
 - static methods, [111](#)
- OracleCollection class
 - constructors, [2](#)
- OracleCollection Class
 - class description, [1](#)
 - properties, [4](#)
 - public methods, [5](#)
- OracleCollectionOptions Class
 - class description, [16](#)
 - properties, [17](#)
- OracleCollectionType Enumeration, [766](#)
- OracleCommand
 - constructors, [21](#)
 - InitialLOBFetchSize property, [132](#)
 - InitialLONGFetchSize property, [131](#)
 - Transaction property, [93](#)
- OracleCommand Class
 - ArrayBindCount, [27](#)
 - class description, [16](#)
 - ExecuteScalar method, [140](#)
 - FetchSize property, [136](#)
 - members, [18](#)
 - properties, [23](#)
 - public methods, [46](#), [15](#)
 - RowSize property, [137](#)
 - static methods, [23](#)
- OracleCommand object, [93](#)
- OracleCommand properties
 - ArrayBindCount, [111](#)
- OracleCommand Transaction object, [94](#)
- OracleCommandBuilder Class, [222](#)
 - class description, [72](#)
 - constructors, [77](#)
 - events, [95](#)
 - members, [74](#)
 - properties, [81](#)
 - public methods, [86](#)
 - static methods, [78](#)
 - updating dataset, [221](#)
- OracleConfiguration
 - members, [98](#)
 - methods, [106](#)
- OracleConfiguration Class
 - class description, [96](#)
 - methods, [105](#)
 - properties, [108](#), [124](#), [142](#), [147](#), [152](#), [161](#), [168](#)
- OracleConnection
 - ClearAllPools property, [55](#)
 - ClearPool property, [55](#)
 - ClientId property, [72](#)
 - events, [313](#)
- OracleConnection class
 - GetSchema methods, [52](#)
- OracleConnection Class
 - class description, [175](#)
 - constructors, [185](#)
 - members, [177](#)
 - obtaining a reference, [167](#)
 - properties, [193](#)
 - static methods, [188](#)
- OracleConnectionOpenEventArgs
 - members, [319](#)
 - properties, [320](#)
- OracleConnectionOpenEventArgs Class, [318](#)
- OracleConnectionOpenEventHandler Delegate,
 - [321](#)
- OracleConnectionOpenReason Enumeration, [766](#)
- OracleConnectionStringBuilder class
 - using, [30](#)
- OracleConnectionStringBuilder Class
 - class description, [322](#)
 - class members, [325](#)
 - constructors, [328](#)
 - public methods, [348](#)
 - public properties, [329](#)
- OracleConnectionType Enumeration
 - description, [767](#)
- OracleCredential, [67](#)
 - constructors, [354](#)
 - members, [353](#)
- OracleCredential Class, [352](#)
 - properties, [357](#)
 - public methods, [360](#)
- OracleCustomTypeMappingAttribute Class
 - constructors, [4](#)
 - description, [1](#)
 - members, [3](#)
 - methods, [6](#)
 - properties, [5](#)
 - static methods, [5](#)
- OracleDataAdapter, [216](#)
 - constructors, [366](#)
 - members, [364](#)
 - SafeMapping Property, [218](#)
 - SelectCommand property, [139](#)
- OracleDataAdapter class
 - FillSchema method, [222](#)
 - Requery property, [220](#)
 - SelectCommand property, [221](#)
- OracleDataAdapter Class, [362](#)
 - events, [383](#)
 - FillSchema method, [222](#)
 - properties, [369](#)
 - public methods, [378](#)
 - SelectCommand property, [222](#)
 - static methods, [369](#)
- OracleDataAdapter Safe Type Mapping, [216](#)
- OracleDatabase Class
 - class description, [387](#)

- OracleDatabase Class (*continued*)
 - constructor, [390](#), [391](#)
 - members, [389](#)
 - properties, [391](#)
 - public methods, [392](#)
 - Shutdown method, [394](#)
 - Startup method, [397](#)
- OracleDatabase Constructors, [390](#)
- OracleDatabaseCharset Enumeration
 - description, [23](#)
- OracleDatabaseCharset enumeration type, [23](#)
- OracleDatabaseNCharset Enumeration
 - description, [29](#)
- OracleDatabaseNCharset enumeration type, [29](#)
- OracleDataProviderInstrumentationOptions Class
 - class description, [1](#)
 - properties, [2](#)
- OracleDataReader, [126](#), [131](#)
 - members, [403](#)
 - retrieving UDTs from, [186](#)
 - typed accessors, [126](#)
- OracleDataReader Class
 - class description, [400](#)
 - FetchSize property, [136](#)
 - populating, [139](#)
 - properties, [408](#)
 - public methods, [422](#)
 - static methods, [407](#)
- OracleDataReader Class SchemaTable, [488](#)
- OracleDataSource
 - members, [511](#)
- OracleDataSource Class
 - class description, [510](#)
 - methods, [512](#)
 - properties, [516](#)
 - Public Events, [520](#)
- OracleDataSource Enumerator class
 - using, [34](#)
- OracleDataSourceBuilder
 - members, [522](#)
- OracleDataSourceBuilder Class
 - class description, [521](#)
 - constructor, [523](#)
 - properties, [527](#)
 - public methods, [523](#)
- OracleDataSourceCollection
 - members, [534](#)
- OracleDataSourceCollection Class
 - class description, [532](#)
 - methods, [535](#)
 - properties, [536](#)
- OracleDataSourceCommand
 - members, [539](#)
- OracleDataSourceCommand Class
 - class description, [538](#)
 - properties, [542](#)
- OracleDataSourceCommand Class (*continued*)
 - public methods, [539](#)
- OracleDataSourceEnumerator Class
 - class description, [543](#)
 - class members, [544](#)
 - public methods, [545](#)
- OracleDate Structure
 - constructors, [69](#)
 - description, [65](#)
 - members, [66](#)
 - methods, [96](#)
 - properties, [91](#)
 - static fields, [73](#)
 - static methods, [75](#)
 - static operators, [82](#)
 - static type conversions, [87](#)
- OracleDBAPrivilege Enumeration
 - description, [768](#)
- OracleDBMSAssert Class
 - class description, [547](#)
 - methods, [547](#)
- OracleDBShutdownMode Enumeration
 - description, [769](#)
- OracleDBStartupMode Enumeration
 - description, [770](#)
- OracleDbType
 - inference, [104](#)
- OracleDbType enumeration, [104](#)
- OracleDbType Enumeration
 - description, [771](#)
- OracleDbType enumeration type, [103](#), [771](#)
- OracleDecimal Structure
 - constructors, [107](#)
 - description, [101](#)
 - instance methods, [173](#)
 - members, [102](#)
 - properties, [168](#)
 - static comparison methods, [118](#)
 - static comparison operators, [150](#)
 - static logarithmic methods, [138](#)
 - static manipulation methods, [123](#)
 - static operators, .NET Type to OracleDecimal, [159](#)
 - static operators, OracleDecimal to .NET, [164](#)
 - static trigonometric methods, [144](#)
- OracleDependency Class
 - change notification, [211](#)
 - class description, [1](#)
 - constructors, [4](#)
 - Continuous Query Notification, [209](#)
 - events, [20](#)
 - instance methods, [17](#)
 - instance properties, [11](#)
 - members, [2](#)
 - static fields, [8](#)
 - static methods, [10](#)

- OracleDRCPYPurity Enumeration
 - description, [772](#)
- OracleDynamicCollection class
 - constructors, [19](#)
- OracleDynamicCollection Class
 - class description, [18](#)
 - properties, [22](#)
 - public methods, [22](#)
- OracleError Class
 - ArrayBindIndex, [561](#)
 - class description, [558](#)
 - members, [560](#)
 - methods, [564](#)
 - properties, [561](#)
 - static methods, [561](#)
- OracleErrorCollection
 - members, [567](#)
 - properties, [568](#)
 - public methods, [569](#)
 - static methods, [568](#)
- OracleErrorCollection Class, [565](#)
- OracleException
 - members, [571](#)
 - methods, [578](#)
 - properties, [573](#)
 - static methods, [572](#)
- OracleException Class, [569](#)
- OracleFailoverEvent Enumeration
 - description, [13](#)
- OracleFailoverEventArgs
 - members, [3](#)
 - properties, [5](#)
 - public methods, [7](#)
- OracleFailoverEventHandler Delegate
 - description, [9](#)
- OracleFailoverRestore Enumeration
 - description, [14](#)
- OracleFailoverReturnCode Enumeration
 - description, [16](#)
- OracleFailoverType Enumeration
 - description, [17](#)
- OracleGlobalization Class
 - class description, [1](#)
 - members, [2](#)
 - properties, [10](#)
 - public methods, [21](#)
- OracleHAEvtArgs Class
 - description, [1](#)
 - members, [2](#)
 - properties, [3](#)
- OracleHAEvtHandler Delegate
 - description, [8](#)
- OracleHAEvtSource Enumeration
 - description, [9](#)
- OracleHAEvtStatus Enumeration
 - description, [10](#)
- OracleIdentityType Enumeration
 - description, [773](#)
- OracleInfoMessageEventArgs
 - members, [582](#)
 - properties, [583](#)
 - public methods, [585](#)
 - static methods, [583](#)
- OracleInfoMessageEventHandler Delegate, [586](#)
- OracleIntervalDS Structure
 - constructors, [184](#)
 - description, [180](#)
 - members, [181](#)
 - methods, [215](#)
 - properties, [210](#)
 - static methods, [191](#)
 - static operators, [199](#)
 - type conversions, [207](#)
- OracleIntervalYM Structure
 - constructors, [223](#)
 - description, [218](#)
 - members, [220](#)
 - methods, [229](#), [250](#)
 - properties, [246](#)
 - static fields, [227](#)
 - static operators, [236](#)
 - type conversions, [244](#)
- OracleNotificationEventArgs Class
 - change notification, [211](#)
 - class description, [28](#)
 - instance methods, [35](#)
- OracleNotificationInfo Enumeration
 - description, [39](#)
- OracleNotificationRequest Class
 - change notification, [211](#)
 - class description, [20](#)
 - Continuous Query Notification, [209](#)
 - instance methods, [27](#)
 - instance properties, [23](#)
 - members, [21](#)
 - static methods, [22](#)
- OracleNotificationSource Enumeration
 - description, [38](#)
- OracleNotificationType Enumeration
 - description, [37](#)
- OracleNullValueException Class
 - class description, [8](#)
 - constructors, [10](#)
 - members, [9](#)
 - methods, [12](#)
 - properties, [12](#)
- OracleObjectMappingAttribute Class
 - constructors, [9](#)
 - description, [7](#)
 - members, [8](#)
 - methods, [13](#)
 - properties, [11](#)

- OracleObjectMappingAttribute Class (*continued*)
 - static methods, [11](#)
- OracleOnsServerCollection
 - members, [595](#)
- OracleOnsServerCollection Class
 - class description, [594](#)
 - methods, [596](#)
 - properties, [598](#)
- OracleOpaqueString
 - members, [600](#)
- OracleOpaqueString Class
 - class description, [599](#)
 - methods, [602](#)
 - properties, [601](#)
- OracleParameter
 - array bind properties, [112](#)
 - ArrayBindSize property, [112](#), [589](#), [591](#), [622](#)
 - ArrayBindStatus property, [112](#), [623](#)
 - constructors, [609](#)
 - inferences of types, [104](#)
 - members, [607](#)
 - properties, [620](#)
 - public methods, [638](#)
 - static methods, [620](#)
- OracleParameter array bind feature, [111](#)
- OracleParameter Class, [605](#)
- OracleParameter object, [102](#)
 - OracleDbType enumerated values, [103](#)
- OracleParameter property
 - ArrayBindSize, [108](#)
 - ArrayBindStatus, [108](#)
 - CollectionType, [108](#)
 - Size, [108](#)
 - Value, [108](#)
- OracleParameterCollection
 - members, [643](#)
 - public methods, [650](#)
 - static methods, [645](#)
- OracleParameterCollection Class, [641](#)
- OracleParameterStatus Enumeration
 - description, [774](#), [776](#)
- OracleParameterStatus enumeration type, [114](#), [774](#), [776](#)
- OraclePasswordAuth Enumeration
 - description, [775](#)
- OraclePasswordAuth enumeration type, [775](#)
- Oraclepermission Class
 - description, [670](#)
- OraclePermission Class
 - constructor, [673](#)
 - members, [671](#)
 - public methods, [674](#)
 - public properties, [674](#)
 - static methods, [673](#)
- OraclePermissionAttribute Class
 - constructor, [680](#)
- OraclePermissionAttribute Class (*continued*)
 - description, [678](#)
 - members, [679](#)
 - public methods, [682](#)
 - public properties, [681](#)
 - static methods, [681](#)
- OraclePoolNameCollection Class
 - class description, [9](#)
 - methods, [10](#)
- OracleRef Class
 - class description, [54](#)
 - class members, [55](#)
 - constructors, [57](#)
 - instance methods, [65](#)
 - instance properties, [61](#)
 - static fields, [59](#)
 - static methods, [60](#)
- OracleRefCursor, [138](#)
- OracleRefCursor Class
 - class description, [163](#)
 - instance methods, [173](#)
 - members, [166](#)
 - populating from a REF CURSOR, [140](#)
 - properties, [168](#)
 - static methods, [167](#), [168](#)
- OracleRefreshAccessTokenEventArgs
 - members, [683](#)
 - properties, [684](#)
- OracleRefreshAccessTokenEventArgs Class
 - class description, [683](#)
- OracleRoleCollection
 - members, [687](#)
- OracleRoleCollection Class
 - class description, [686](#)
 - methods, [687](#)
 - properties, [693](#)
- OracleRowsCopiedEventArgs Class
 - class description, [47](#)
 - constructors, [49](#)
 - members, [48](#)
 - properties, [50](#)
- OracleRowsCopiedEventHandler Delegate
 - description, [46](#)
- OracleRowUpdatedEventArgs
 - constructor, [698](#)
 - members, [696](#)
 - properties, [699](#)
 - public methods, [700](#)
 - static methods, [698](#)
- OracleRowUpdatedEventArgs Class, [695](#)
- OracleRowUpdatedEventHandler Delegate, [700](#)
- OracleRowUpdatingEventArgs
 - constructor, [704](#)
 - members, [702](#)
 - properties, [705](#)
 - public methods, [706](#)

- OracleRowUpdatingEventArgs (*continued*)
 - static methods, [704](#)
- OracleRowUpdatingEventArgs Class, [701](#)
- OracleRowUpdatingEventHandler Delegate, [706](#)
- OracleServiceCollectionExtensions Class
 - class description, [34](#)
 - static methods, [34](#)
- OracleSessionlessTransactionOption Class
 - class description, [707](#)
- OracleSessionlessTransactionOptions
 - constructors, [709](#)
 - members, [708](#)
- OracleSessionlessTransactionOptions Class
 - properties, [711](#)
- OracleSessionlessTransactionStartBehavior
 - Enumeration
 - description, [776](#)
- OracleSessionlessTransactionStartBehavior
 - enumeration type, [776](#)
- OracleShardingKey
 - constructors, [715](#)
 - members, [715](#)
- OracleShardingKey Class, [713](#)
 - instance methods, [717](#)
- OracleSparseVector
 - members, [719](#)
- OracleSparseVector Class
 - class description, [718](#)
 - instance methods, [731](#)
 - properties, [726](#)
- OracleString Structure
 - constructors, [257](#)
 - description, [253](#)
 - members, [254](#)
 - methods, [278](#)
 - properties, [275](#)
 - static fields, [262](#)
 - static methods, [263](#)
 - static operators, [268](#)
 - type conversions, [274](#)
- OracleTimeStamp Structure
 - constructors, [289](#)
 - description, [283](#)
 - members, [285](#)
 - methods, [327](#)
 - properties, [321](#)
 - static methods, [297](#)
 - static operators, [305](#)
 - static type conversions, [315](#)
- OracleTimeStampLTZ Structure
 - constructors, [345](#)
 - description, [339](#)
 - members, [341](#)
 - methods, [385](#)
 - properties, [379](#)
 - static fields, [352](#)
- OracleTimeStampLTZ Structure (*continued*)
 - static methods, [354](#)
 - static operators, [362](#)
 - static type conversions, [373](#)
- OracleTimeStampTZ Structure
 - constructors, [403](#)
 - description, [397](#)
 - members, [399](#)
 - methods, [449](#)
 - properties, [443](#)
 - static fields, [416](#)
 - static methods, [418](#)
 - static operators, [426](#)
 - static type conversions, [436](#)
- OracleTransaction
 - members, [739](#)
 - properties, [740](#)
 - public methods, [742](#)
 - static methods, [740](#)
- OracleTransaction Class
 - class description, [736](#)
- OracleTruncateException Class
 - class description, [13](#)
 - constructors, [15](#)
 - members, [14](#)
 - methods, [17](#)
 - properties, [17](#)
 - static methods, [16](#)
- OracleTxEventQNotificationDirective Enumeration
 - description, [93](#)
- OracleTypeException Class
 - class description, [1](#)
 - constructors, [3](#)
 - members, [2](#)
 - properties, [5](#)
 - static methods, [5](#)
- OracleUdt Class
 - description, [28](#)
 - members, [29](#)
 - static methods, [29](#)
- OracleUdtFetchOption Enumeration
 - description, [77](#)
- OracleUdtStatus Enumeration
 - description, [77](#)
- OracleVector
 - members, [751](#)
- OracleVector Class
 - class description, [750](#)
 - instance methods, [761](#)
 - properties, [757](#)
- OracleVectorStore Class
 - class description, [43](#)
 - public methods, [46](#)
- OracleVectorStoreOptions Class
 - class description, [51](#), [52](#)
 - properties, [52](#)

OracleVectorType Enumeration
 description, [778](#)
 OracleVectorType enumeration type, [778](#)
 OracleXmlCommandType Enumeration, [1](#)
 OracleXmlQueryProperties Class
 class description, [2](#)
 constructors, [6](#)
 members, [5](#)
 properties, [7](#)
 public methods, [10](#)
 OracleXmlSaveProperties Class, [11](#)
 constructors, [15](#)
 members, [14](#)
 properties, [15](#)
 public methods, [19](#)
 OracleXmlStream Class
 class description, [20](#)
 constructors, [23](#)
 instance methods, [28](#)
 instance properties, [24](#)
 members, [21](#)
 static methods, [23](#)
 OracleXmlType Class, [167](#)
 class description, [39](#)
 constructors, [42](#)
 instance methods, [53](#)
 instance properties, [47](#)
 members, [40](#)
 static methods, [46](#)
 outages, [78](#)

P

parameter binding, [102](#)
 parameter binding with OracleParameter, [188](#)
 password expiration, [69](#)
 passwords in code examples, [ii](#)
 performance, [74](#), [114](#)
 array binding, [111](#)
 connection pooling, [53](#)
 fine-tuning FetchSize, [137](#)
 number of rows fetched, [136](#)
 Obtaining LOB Data, [132](#)
 performance counters
 connection pool, [56](#)
 publishing, [56](#), [59](#)
 PL/SQL Associative Array binding, [108](#)
 PL/SQL Index-By Tables, [108](#)
 PL/SQL language, [138](#)
 PL/SQL REF CURSOR, [138](#)
 PL/SQL REF CURSOR and OracleRefCursor,
 [138](#)
 planned outage, [78](#)
 PLSQLAssociativeArray, [766](#)
 pool size attributes
 Oracle RAC, [80](#)

populating an OracleDataReader from a REF
 CURSOR, [139](#)
 populating an OracleRefCursor from a REF
 CURSOR, [140](#)
 populating the DataSet from a REF CURSOR,
 [139](#)
 populating the DataSet with generic and custom
 objects, [191](#)
 port
 listen for database notifications, [77](#)
 port number
 defining listener, [212](#)
 preventing data loss, [216](#), [218](#)
 preventing logical corruption, [80](#)
 PrimaryKey property, [221](#)
 configuring, [222](#)
 privileged connections, [66](#)
 promotable transactions, [95](#)
 properties
 ClientId property, [72](#)
 provider factory classes, [2](#)
 proxy authentication, [71](#)

Q

query result set
 retrieving as XML, [169](#)

R

REF CURSOR, [142](#)
 behavior of ExecuteScalar method, [140](#)
 implicit, [142](#)
 obtaining, [139](#)
 passing to stored procedure, [141](#)
 populating DataSet from, [139](#)
 populating from OracleDataReader, [139](#)
 registry entries, [11](#)
 release Oracle8i (8.1.7), [174](#)
 release Oracle9i(9.0.x), [174](#)
 Requery property, [220](#)
 round-trip, [111](#)
 RowSize property, [137](#)
 runtime connection load balancing, [76](#)

S

Safe Type Mapping, [216](#)
 SafeMapping Property, [218](#)
 samples, [6](#), [9](#)
 UDT, [1](#)
 saving change using an XML document, [176](#)
 saving changes
 using XML data, [174](#)
 schema metadata
 customizing metadata, [52](#)

SchemaTable, [488](#)
 search order
 unmanaged DLLs, [7](#)
 SecureFiles, [150](#)
 SelectCommand property, [139](#)
 session globalization parameters, [226](#)
 session globalization settings, [224](#)
 Shutdown method, [394](#)
 simple application, [22](#)
 Size property, [108](#)
 Startup method, [397](#)
 Statement Caching
 connection string attributes, [115](#)
 methods and properties, [115](#)
 Statement Cache Purge, [114](#)
 Statement Cache Size, [114](#)
 StatementCacheWithUdts, [192](#)
 stored procedures and functions, [141](#)
 Stream class, [151](#)
 SYSDBA privileges, [66](#)
 SYSOPER privileges, [66](#)
 system requirements
 Oracle Data Provider for .NET, [1](#)
 System.Transactions support, [95](#)

T

table or view
 special characters in, [175](#)
 Temporary LOBs, [153](#)
 thread globalization settings, [226](#)
 thread-based globalization settings, [224](#)
 Thread.Abort method, [78](#)
 tips for ODP.NET, [78](#)
 TNS alias, [30](#)
 tracing attributes, [72](#)
 Transaction Guard, [80](#)
 Transaction object, [94](#)
 Transaction property, [93](#)
 transactions
 commit, [177](#)
 TransactionScope, [95](#)
 triggers, [198](#)
 insert triggers, [198](#)
 troubleshooting, [237](#)
 typed OracleDataReader accessors, [126](#)

U

UDT
 method invocation, [192](#)
 UDT metadata retrieval from OracleDataReader,
 [187](#)
 UdtCacheSize, [192](#)
 UDTs, [178](#)
 collection types, [178](#)

UDTs (*continued*)
 configuration settings, [192](#)
 object types, [178](#)
 parameter binding with OracleParameter, [188](#)
 retrieving from OracleDataReader, [186](#)
 samples, [1](#)
 UDTs (Oracle User-Defined Types), [178](#)
 UdtTypeName property, [188](#)
 unique columns, [131](#), [132](#)
 unique constraint, [131](#), [132](#)
 unique index, [131](#), [132](#)
 UniqueConstraint, [222](#)
 uniqueness
 in updating DataSet to database, [221](#)
 uniqueness in DataRows, [221](#)
 unmanaged DLLs
 mismatch, [7](#)
 search order, [7](#)
 unmanged DLLs
 config support, [7](#)
 updating
 LOBs, [152](#)
 updating a DataSet obtained from a REF
 CURSOR, [140](#)
 updating LOBs using a DataSet, [152](#)
 updating LOBs using ODP.NET LOB objects, [153](#)
 updating LOBs using OracleCommand and
 OracleParameter, [152](#)
 updating without PrimaryKey and Constraints, [222](#)
 user-defined types, [178](#)
 UserCallCompleted public read-only property, [591](#)
 using FetchSize property, [136](#)

V

Value property, [108](#)
 VARRAY types, [178](#)
 Virtual Private Database(VPD), [72](#)
 Visual Studio
 documentation, [6](#), [7](#)

W

web.config, [11](#)
 Windows registry, [25](#)

X

XML
 characters with special meaning, [169](#)
 data manipulation using, [173](#)
 XML data
 saving changes using, [174](#)
 updating in OracleXmlType, [169](#)
 XML Database, [165](#)
 XML DB, [165](#), [166](#)

- XML element name
 - case-sensitivity in, [175](#)
- XML Element Name to Column Name Mapping, [175](#)
- XML related classes, [1](#)
- XML related enumerations, [1](#)
- XML Support, [165](#)
- XML to specify custom type mappings, [182](#)
- XMLQuery, [166](#)
- XMLTable, [166](#)
- XMLType column
 - as a .NET String, [168](#)
 - fetching into the DataSet, [168](#)
 - updating with OracleCommand, [168](#)
- XMLType columns
 - setting to NULL, [168](#)
- XQuery
 - support, [166](#)
- XQUERY, [166](#)
- XQuery language, [166](#)