

iWay

iWay Data Adapter Administration for UNIX,
Windows, OpenVMS, OS/400, OS/390 and z/OS
Version 5 Release 2.3

EDA, EDA/SQL, FIDEL, FOCCALC, FOCUS, FOCUS Fusion, FOCUS Vision, Hospital-Trac, Information Builders, the Information Builders logo, Parlay, PC/FOCUS, SmartMart, SmartMode, SNAPPack, TableTalk, WALDO, Web390, WebFOCUS and WorldMART are registered trademarks, and iWay and iWay Software are trademarks of Information Builders, Inc.

Due to the nature of this material, this document refers to numerous hardware and software products by their trademarks. In most, if not all cases, these designations are claimed as trademarks or registered trademarks by their respective companies. It is not this publisher's intent to use any of these names generically. The reader is therefore cautioned to investigate all claimed trademark rights before using any of these names other than to refer to the product described.

Copyright © 2003, by Information Builders, Inc. All rights reserved. This manual, or parts thereof, may not be reproduced in any form without the written permission of Information Builders, Inc.

Printed in the U.S.A.

Preface

This manual describes configuration and management of Data Adapters.

Documentation Conventions

The following conventions apply throughout this manual:

Convention	Description
THIS TYPEFACE or <i>this typeface</i>	Denotes syntax that you must enter exactly as shown.
<i>this typeface</i>	Represents a placeholder (or variable) in syntax for a value that you or the system must supply.
<u>underscore</u>	Indicates a default setting.
<i>this typeface</i>	Represents a placeholder (or variable), a cross-reference, or an important term. It may also indicate a button, menu item, or dialog box option you can click or select.
this typeface	Highlights a file name or command.
Key + Key	Indicates keys that you must press simultaneously.
{ }	Indicates two or three choices; type one of them, not the braces.
[]	Indicates a group of optional parameters. None are required, but you may select one of them. Type only the parameter in the brackets, not the brackets.
	Separates mutually exclusive choices in syntax. Type one of them, not the symbol.
...	Indicates that you can enter a parameter multiple times. Type only the parameter, not the ellipsis points (...).
.	Indicates that there are (or could be) intervening or additional commands.

Related Publications

To view a current listing of our publications and to place an order, visit our World Wide Web site, <http://www.iwaysoftware.com>. You can also contact the Publications Order Department at (800) 969-4636.

Customer Support

Do you have questions about iWay Data Adapters?

Call Information Builders Customer Support Service (CSS) at (800) 736-6130 or (212) 736-6130. Customer Support Consultants are available Monday through Friday between 8:00 a.m. and 8:00 p.m. EST to address all your iWay Data Adapters questions. Information Builders consultants can also give you general guidance regarding product capabilities and documentation. Please be ready to provide your six-digit site code (xxxx.xx) when you call.

You can also access support services electronically, 24 hours a day, with InfoResponse Online. InfoResponse Online is accessible through our World Wide Web site, <http://www.informationbuilders.com>. It connects you to the tracking system and known-problem database at the Information Builders support center. Registered users can open, update, and view the status of cases in the tracking system and read descriptions of reported software issues. New users can register immediately for this service. The technical support section of www.informationbuilders.com also provides usage techniques, diagnostic tips, and answers to frequently asked questions.

To learn about the full range of available support services, ask your Information Builders representative about InfoResponse Online, or call (800) 969-INFO.

Information You Should Have

To help our consultants answer your questions most effectively, be ready to provide the following information when you call:

- Your six-digit site code (xxxx.xx).
- Your iWay Software configuration:
 - The iWay Software version and release. You can find your server version and release using the *Version* option in the Web Console. (**Note:** the MVS and VM servers do not use the Web Console.)
 - The communications protocol (for example, TCP/IP or LU6.2), including vendor and release.
- The stored procedure (preferably with line numbers) or SQL statements being used in server access.
- The database server release level.

- The database name and release level.
- The Master File and Access File.
- The exact nature of the problem:
 - Are the results or the format incorrect? Are the text or calculations missing or misplaced?
 - The error message and return code, if applicable.
 - Is this related to any other problem?
- Has the procedure or query ever worked in its present form? Has it been changed recently? How often does the problem occur?
- What release of the operating system are you using? Has it, your security system, communications protocol, or front-end software changed?
- Is this problem reproducible? If so, how?
- Have you tried to reproduce your problem in the simplest form possible? For example, if you are having problems joining two data sources, have you tried executing a query containing just the code to access the data source?
- Do you have a trace file?
- How is the problem affecting your business? Is it halting development or production? Do you just have questions about functionality or documentation?

User Feedback

In an effort to produce effective documentation, the Documentation Services staff welcomes your opinions regarding this manual. Please use the Reader Comments form at the end of this manual to relay suggestions for improving the publication or to alert us to corrections. You can also use the Documentation Feedback form on our Web site, <http://www.iwaysoftware.com>

Thank you, in advance, for your comments.

iWay Software Training and Professional Services

Interested in training? Our Education Department offers a wide variety of training courses for iWay Software and other Information Builders products.

For information on course descriptions, locations, and dates, or to register for classes, visit our World Wide Web site (<http://www.iwaysoftware.com>) or call (800) 969-INFO to speak to an Education Representative.

Interested in technical assistance for your implementation? Our Professional Services department provides expert design, systems architecture, implementation, and project management services for all your business integration projects. For information, visit our World Wide Web site (<http://www.iwaysoftware.com>).

Contents

1. Introduction to iWay Data Adapters	1-1
Functions of a Data Adapter	1-2
How a Data Adapter Works	1-2
Processing SQL Requests	1-3
Relational and Non-Relational Data Adapters	1-4
Supported Data Sources	1-5
Data Management	1-6
Describing Data Sources	1-6
Processing Requests	1-7
Master File	1-8
Access File	1-11
Primary Key	1-13
Creating Virtual Fields	1-13
Cross-Century Dates	1-14
Cross-Century Dates SET Commands	1-15
Master File Syntax	1-16
Metadata Services With SQLENGINE SET	1-17
How Applications Access Metadata	1-17
Obtaining Column Information (DB2 only)	1-18
Obtaining a User-Defined Metadata	1-18
Maintaining Upward Compatibility	1-21
Additional Master File Attributes	1-21
Documenting a Table	1-22
Documenting a Column	1-22
Supplying an Alternate Column Title	1-23
Specifying an Online Help Message	1-25
When a Help Message Displays	1-25

2. Getting Started in Adabas	2-7
Preparing the Adabas Environment	2-8
Configuring the Data Adapter for Adabas	2-9
Declaring Connection Attributes	2-10
Overriding Default Passwords in Specific Files	2-10
Adabas Overview	2-11
Adabas Files	2-14
Inverted Lists: The Adabas Key Structure	2-14
Field Definition Tables	2-15
Managing Data Storage With Null-Suppression	2-17
Server File Structure	2-18
Adabas Descriptors	2-19
Adabas Files With Fixed-length Records	2-20
Adabas Files With Variable-length Records and Repeating Fields	2-21
Managing Adabas Metadata	2-21
Creating Synonyms	2-22
Overview of Master and Access Files	2-35
Master Files for Adabas	2-35
File Attributes in Master Files	2-36
Segment Attributes in Master Files	2-37
Field Attributes in Master Files	2-40
Access Files for Adabas	2-45
Release Declaration	2-46
Segment Attributes in Access Files	2-47
Mapping Adabas Descriptors	2-54
Specifying INDEX=I	2-54
Describing Superdescriptors	2-54
Specifying Superdescriptors Using the GROUP Attribute	2-55
Calculating GROUP Length	2-55
Describing Descriptors in the Access File	2-57
Specifying Superdescriptors Containing Partial Fields	2-58
Specifying Subdescriptors	2-58
Specifying Phonetic Descriptors	2-58
Specifying Hyper Descriptors	2-58
Specifying Null-Suppression	2-59
Implementing Embedded JOINS: KEYFLD and IXFLD	2-59
Mapping Adabas Files With Variable-length Records and Repeating Fields	2-62
Describing Multi-value Fields and Periodic Groups With the OCCURS Attribute	2-65
Describing Multi-value Fields Within Periodic Groups	2-67
Specifying OCCURS Segment Sequence: The ORDER Field	2-69
Using the GROUP Attribute to Cross-Reference Files	2-70
Platform-Specific Functionality	2-71

Customizing the Adabas Environment	2-72
ORDER Fields in the Indexed Field List	2-72
Switching the Access Mode Using NOPACCESS	2-72
NEW Synonym Setting for Superdescriptors	2-72
Multifetch and Prefetch	2-74
Adabas Dynamic Database Number	2-76
Running in 24-Bit Mode	2-77
Adabas Reporting Considerations	2-78
Adabas File Navigation Techniques	2-78
Referencing Subtrees and Record Retrieval	2-78
Segment Retrieval Methodology	2-78
Missing Unique Segments	2-79
Adabas Selection Considerations	2-82
Selection Order in the Access File	2-82
Using the JOIN Command to Process Multiple Files	2-85
Adabas Optimization With Null-Suppression for CALLTYPE=RL	2-86
Adabas Optimization on Group Fields	2-87
Test on Group Field With Numerics	2-87
Adabas Writing Considerations	2-88
Adabas Write Data Adapter	2-88
SQL Commands for Data Adapter for Adabas	2-89
Data Adapter Navigation	2-94
Entry Segment Retrieval of Adabas Records	2-95
Read Physical Calls	2-96
Read Logical Navigation	2-97
Read Logical Calls Without a Starting Value	2-98
Read Logical Calls With a Starting Value	2-98
Retrieval Through Read Logical by ISN (L1) Calls	2-99
FIND Navigation	2-102
Simple FIND Calls	2-102
Complex FIND Calls	2-103
Read Descriptor Value (L9) Direct Calls	2-104
Descendant Periodic Groups and Multi-value Fields	2-105
Descendant Adabas Records	2-105
Read Logical Calls Using a Starting Value	2-106
Simple FIND Calls (Descendant Records)	2-106
Complex FIND Calls (Descendant Records)	2-107

3.	Getting Started in Allbase	3-1
	Preparing the Allbase Environment	3-2
	Configuring the Data Adapter for Allbase	3-2
	Declaring Connection Attributes	3-2
	Overriding the Default Connection	3-3
	Controlling the Connection Scope	3-4
	Managing Allbase Metadata	3-4
	Creating Synonyms	3-5
	Data Type Support	3-11
	CLOB Activation	3-12
	Customizing the Allbase Environment	3-13
	Obtaining the Number of Rows Updated or Deleted	3-13
4.	Getting Started in Bull GCOS	4-1
	Preparing the Bull GCOS Environment	4-2
	Operations Considerations	4-2
	Configuration Considerations	4-2
	Functional Limitation Considerations	4-3
	Configuring the Data Adapter for Bull GCOS	4-4
	Declaring Connection Attributes	4-7
	Controlling the Connection Scope	4-9
	Managing Bull GCOS Metadata	4-9
	Creating Synonyms	4-10
	Customizing the Bull GCOS Environment	4-14
	Establishing an FTP Connection to an Intermediate Server	4-14
	Specifying a Timeout Limit	4-15
	Setting the Working Directory	4-15
	Displaying Current Adapter Settings	4-16
	Enabling iWay for Bull Run-Time Environment (RTE)	4-17
5.	Getting Started in Informix C-ISAM, Micro Focus C-ISAM, VSAM, and Flat Files	5-1
	Preparing the C-ISAM Environment	5-2
	Specifying the Location of C-ISAM Files	5-2
6.	Getting Started in DATACOM	6-1
	Preparing the DATACOM Environment	6-2
	Configuring the Data Adapter for DATACOM	6-2
	Declaring Connection Attributes	6-2
	User Requirements Table	6-2
	Element-level Security	6-2
	DATACOM Overview and Mapping Considerations	6-2
	Data Structures	6-3
	Mapping Structures in the Server	6-5
	Managing DATACOM Metadata	6-8

Master Files for DATACOM	6-9
File Attributes	6-9
Segment Attributes	6-10
Field Attributes	6-10
Access Files for DATACOM	6-12
Describing Multi-file Structures for DATACOM	6-14
Multi-file Master File	6-15
Multi-file Access File	6-16
Using Multiple Fields to Cross-reference Data Sources	6-17
DATACOM Data Dictionary Master and Access Files	6-18
Data Dictionary PERSON-MASTER Indented Report	6-18
Data Dictionary Element Field Report for EMDTA	6-19
PERSON Master File and Access File	6-19
Data Dictionary PAYROLL-MASTER Indented Report	6-20
Data Dictionary Element Field Report for PAYROLL	6-20
PAYROLL Master File and Access File	6-21
PERSPAY Master File and Access File	6-21
Data Retrieval Logic for DATACOM	6-22
GSETL and GETIT	6-22
SELFR and SELNR	6-22
7. Getting Started in DB2	7-1
Preparing the DB2 Environment	7-2
Accessing a Remote Database Server	7-3
XA Support	7-3
Configuring the Data Adapter for DB2	7-3
Declaring Connection Attributes	7-4
DB2 CURRENT SQLID (OS/390 and z/OS)	7-7
Overriding the Default Connection	7-8
Controlling Connection Scope (OS/390 and z/OS)	7-9
Managing DB2 Metadata	7-11
Creating Synonyms	7-11
Data Type Support	7-17
Controlling the Mapping of Large Character Data Types	7-18
Controlling the Mapping of Variable-Length Data Types	7-19
BLOB Activation (OS/390 and z/OS)	7-20
BLOB Read/Write Support (OS/390 and z/OS)	7-21
Changing the Precision and Scale of Numeric Columns	7-22

Customizing the DB2 Environment	7-24
Improving Response Time	7-24
Designating a Default Tablespace	7-25
Specifying the Block Size for Array Retrieval	7-26
Controlling the Types of Locks	7-27
Overriding Default Parameters for Index Space	7-28
Obtaining the Number of Rows Updated or Deleted	7-29
Using DB2 Aliases	7-29
Calling a DB2 Stored Procedure Using SQL Passthru	7-30
8. Getting Started in Enterprise Java Beans	8-1
Preparing the Web Application Server Environment	8-2
Configuring the Data Adapter for Enterprise Java Beans	8-2
Declaring Connection Attributes for the Web Application Server	8-2
Selecting a Web Application Server to Access	8-5
Controlling Connection Scope	8-5
Managing Enterprise Java Beans Metadata	8-6
Creating Synonyms	8-6
Data Type Support	8-11
9. Getting Started in Essbase	9-1
Preparing the Essbase Environment	9-2
Configuring the Data Adapter for Essbase	9-2
Declaring Connection Attributes	9-2
Managing Essbase Metadata	9-4
Creating Synonyms	9-4
Parent/Child Support	9-7
Support for User-Defined Attributes (UDAs)	9-8
Describing Scenario Dimensions in the Master File	9-9
Describing the Measures Dimension in the Master File	9-10
Changing the Default Usage Format of the Accounts Dimension	9-12
Customizing the Essbase Environment	9-12
Adapter Functionality	9-12
Specifying ALIAS Names	9-15
Specifying ALIAS Tables	9-16
Setting the Maximum Number of Rows Returned	9-17
Time Series Reporting	9-17
Summing on Non-Aggregated Fields	9-18
Preventing Aggregation of Non-Consolidating Members	9-19
Suppressing Shared Members	9-20
Suppressing Zero Values	9-22
Suppressing Missing Data	9-22
Suppressing Zero Values and Missing Data	9-23
Substitution Variables	9-24

10. Getting Started in CA-IDMS/DB	10-1
Preparing the IDMS/DB Environment	10-2
Configuring the Data Adapter for IDMS/DB	10-2
Declaring Connection Attributes	10-2
IDMS/DB Overview and Mapping Considerations	10-3
Mapping Concepts	10-3
Set-Based Relationships	10-5
Simple Set	10-6
Common Owner	10-7
Common Member	10-8
Multi-Member	10-9
Bill-of-Materials	10-10
Loop Structures	10-12
CALC-Based and Index-Based Relationships	10-13
Logical Record Facility Concepts	10-14
LRF Records as Descendants	10-15
Summary of Network Relationships	10-16
Managing IDMS/DB Metadata	10-17
Master Files for IDMS/DB	10-17
File Attributes	10-17
Segment Attributes	10-18
Field Attributes	10-19
Remote Descriptions	10-25
Intra-record Structures: OCCURS Segment	10-26
Describing the Repeating Group to the Server	10-26
Access Files for IDMS/DB	10-31
CA-IDMS/DB Access File Syntax	10-32
Subschema Declaration Keywords	10-32
CV Mode Only	10-33
Segment Declaration Keywords for Network Record Types	10-34
Segment Declaration Keywords for LRF Records	10-37
Index Declaration Keywords for Network Record Types	10-39
IDMS/DB Sample File Descriptions	10-40
Schema: EMPSCHM	10-40
Network Subschema: EMPSS01	10-48
Master File for Network	10-49
Access File for Network	10-54
LRF Subschema: EMPSS02	10-55
Master File for LRF	10-58
Access File for LRF	10-59
Sample of a Partial LRF Record	10-59
SPF Indexes	10-60

Customizing the IDMS/DB Environment	10-61
Retrieval Subtree	10-62
Retrieval Sequence	10-63
Retrieval Sequence With Unique Segments	10-64
Effects of Screening Conditions on Retrieval	10-65
Restricting the Number of Records Retrieved	10-67
Screening Conditions With Unique Segments	10-68
Retrieving Short Paths	10-68
Short Paths in Unique Descendants	10-69
Short Paths in Non-Unique Descendants	10-69
Selective ALL Prefix	10-72
File Inversion	10-73
Joining Master Files	10-75
Data Adapter for IDMS/DB Navigation	10-78
IDMS/DB File Retrieval	10-78
Retrieval Subtree	10-79
Retrieval Sequence With Unique Segments	10-80
Screening Conditions	10-81
Screening Conditions With Unique Segments	10-82
Short Paths	10-82
Short Paths in Unique Descendants	10-82
Short Paths in Non-Unique Descendants	10-83
CA-IDMS/DB Record Retrieval Internals	10-83
Entry Segment Retrieval of Network Records	10-83
Retrieval by Database Key	10-84
Retrieval by CALC Field	10-84
Retrieval by Index	10-85
SEQFIELD Parameter	10-86
Retrieval by Area Sweep	10-87
Descendant Segment Retrieval of Network Records	10-87
Set-Based Retrieval	10-88
CALC-Based Retrieval	10-89
Index-Based Retrieval	10-89
LRF Record Retrieval	10-90
Overriding DBNAME and DICTNAME	10-91
Tracing the Data Adapter for IDMS/DB	10-92
11. Getting Started in CA-IDMS/SQL	11-1
Preparing the IDMS/SQL Environment	11-2
Configuring the Data Adapter for IDMS/SQL	11-2
Overriding the Default Connection	11-2
Other Session Commands	11-3
Controlling the Connection Scope	11-3

Managing IDMS/SQL Metadata	11-3
Creating Synonyms	11-4
Master Files	11-5
Access Files	11-9
Primary Key	11-11
Data Type Support	11-12
Changing the Precision and Scale of Numeric Columns	11-13
Customizing the IDMS/SQL Environment	11-15
Setting the User-specific Schema Name	11-15
Controlling Transactions	11-15
Designating a Default Tablespace	11-17
Overriding Default Parameters for Index Space	11-17
Obtaining the Number of Rows Updated or Deleted	11-18
12. Getting Started in IMS	12-1
Preparing the IMS Environment	12-2
Establishing Security	12-7
Configuring the Data Adapter for IMS	12-9
Dynamic Setting of PSB	12-9
Managing IMS Metadata	12-10
Creating Synonyms	12-10
Migrating From an Existing MVS Server	12-17
13. Getting Started in Informix	13-1
Preparing the Informix Environment	13-2
Accessing a Remote Informix Server	13-3
Informix SQL ID (for Informix SE only)	13-3
ANSI-Compliant Data Sources	13-3
XA Support	13-4
Configuring the Data Adapter for Informix	13-4
Declaring Connection Attributes	13-4
Overriding the Default Connection	13-7
Controlling the Connection Scope	13-8
Managing Informix Metadata	13-8
Creating Synonyms	13-8
Data Type Support	13-14
Controlling the Mapping of Large Character Data Types	13-16
Controlling the Mapping of Variable-Length Data Types	13-17
Changing the Precision and Scale of Numeric Columns	13-18
Customizing the Informix Environment	13-20
Obtaining the Number of Rows Updated or Deleted	13-20
Calling an Informix Stored Procedure Using SQL Passthru	13-21

14. Getting Started in Ingres II	14-1
Preparing the Ingres II Environment	14-2
Configuring the Data Adapter for Ingres II	14-2
Declaring Connection Attributes	14-2
Overriding the Default Connection	14-3
Managing Ingres II Metadata	14-4
Creating Synonyms	14-4
Data Type Support	14-10
15. Getting Started in Interplex	15-1
Preparing the Interplex Environment	15-2
Configuring the Data Adapter for Interplex	15-2
Declaring Connection Attributes	15-2
Overriding the Default Connection	15-4
Controlling the Connection Scope	15-5
Managing Interplex Metadata	15-5
Creating Synonyms	15-5
Data Type Support	15-10
Changing the Precision and Scale of Numeric Columns	15-11
Customizing the Interplex Environment	15-13
Obtaining the Number of Rows Updated or Deleted	15-13
16. Getting Started in J.D. Edwards Query Adapter	16-1
Overview of the Setup Process	16-2
Installation Prerequisites	16-2
Unloading the CD-ROM	16-3
Working With QSYS and IFS File Systems	16-4
Copying Files	16-4
Linking Files	16-5
Using EDAPATH	16-5
Using APP	16-6
Generating Metadata	16-6
Using AUTOSQL to Generate Master and Access Files	16-7
Using AUTO400 to Generate Master and Access Files	16-11
Using JDECONV	16-16
Using Master Files and Access Files	16-21
Enabling J.D. Edwards Security	16-22
Using the Report Library	16-23
Enabling Tracing	16-23
J.D. Edward Major System Codes	16-24
Frequently Asked Questions	16-27

17. Getting Started in Lawson	17-1
Preparing the Server Environment for Adapter Configuration	17-2
Configuring the Adapter for Lawson	17-4
Managing Metadata	17-5
Using the Query Adapter for Lawson	17-6
18. Getting Started in Microsoft Access	18-1
Preparing the Microsoft Access Environment	18-2
Configuring the Data Adapter for Microsoft Access	18-2
Declaring Connection Attributes	18-2
Overriding the Default Connection	18-4
Controlling the Connection Scope	18-4
Managing Microsoft Access Metadata	18-5
Creating Synonyms	18-5
Data Type Support	18-11
Controlling the Mapping of Large Character Data Types	18-13
Changing the Precision and Scale of Numeric Columns	18-14
Customizing the Microsoft Access Environment	18-17
Obtaining the Number of Rows Updated or Deleted	18-17
19. Getting Started in Microsoft SQL Server	19-1
Preparing the Microsoft SQL Server Environment	19-2
Accessing Microsoft SQL Server Remotely	19-2
XA Support	19-2
Configuring the Data Adapter for Microsoft SQL Server	19-3
Declaring Connection Attributes	19-3
Overriding the Default Connection	19-7
Controlling the Connection Scope	19-8
Managing Microsoft SQL Server Metadata	19-8
Creating Synonyms	19-8
Data Type Support	19-14
Controlling the Mapping of Large Character Data Types	19-16
Controlling the Mapping of Variable-Length Data Types	19-18
Enabling National Language Support	19-19
Changing the Precision and Scale of Numeric Columns	19-20
Support of Read-Only Fields	19-22
Customizing the Microsoft SQL Server Environment	19-23
Specifying a Timeout Limit	19-23
Specifying the Cursor Type	19-24
Specifying the Block Size for Array Retrieval	19-25
Specifying the Login Wait Time	19-26
Activating NONBLOCK Mode	19-27
Obtaining the Number of Rows Updated or Deleted	19-28
Controlling Transactions	19-29

Calling a Microsoft SQL Server Stored Procedure Using SQL Passthru	19-30
Microsoft SQL Server Compatibility With ODBC	19-31
Microsoft SQL Server Connection Attributes With ODBC	19-31
Microsoft SQL Server Cursors With ODBC	19-32
Mapping of UNIQUEIDENTIFIER and BIT Data Types	19-32
20. Getting Started in Microsoft SQL Server Analytical Engine	20-1
Preparing the Microsoft SQL Server Analytical Engine Environment	20-2
Setting Microsoft SQL Server Analytical Engine Security	20-2
Accessing a Microsoft SQL Server Analytical Engine Server	20-3
Configuring the Data Adapter for Microsoft SQL Server Analytical Engine	20-3
Declaring Connection Attributes	20-3
Changing the Default Connection	20-4
Managing Microsoft SQL Server Analytical Engine Metadata	20-6
Creating Synonyms	20-6
Customizing the Microsoft SQL Server Analytical Engine Environment	20-8
Activating NONBLOCK Mode and Issuing a TIMEOUT Limit	20-8
Accelerating Queries	20-9
21. Getting Started in MODEL 204	21-1
Preparing the MODEL 204 Environment	21-2
Configuring the Data Adapter for MODEL 204	21-2
Specifying Account and File Passwords	21-2
Testing the Adapter Installation	21-3
MODEL 204 Overview and Mapping Considerations	21-3
Files With One Logical Record Type	21-4
Files With Multiply Occurring Fields	21-5
Describing Files to the Server	21-5
Mapping MODEL 204 and Server Relationships	21-6
Dynamic Joins	21-7
Embedded Joins	21-9
Joining Unrelated Files	21-12
Summary of Mapping Rules	21-13
Managing MODEL 204 Metadata	21-14
Master Files	21-14
Access Files	21-21

Customizing the MODEL 204 Environment	21-31
Specifying a User Account and Password	21-32
Specifying the Capacity of Adapter Buffers	21-33
Controlling the Size of the FBTL Buffer	21-34
Indicating Missing Data on Reports	21-34
Locking Records to Ensure Accurate Data Reports	21-35
Controlling Thread Management	21-36
Displaying Adapter Defaults and Current Settings	21-36
Adapter Tracing for MODEL 204	21-36
22. Getting Started in MQSeries	22-1
Preparing the MQSeries Environment	22-2
Configuring the Data Adapter for MQSeries	22-2
Declaring Connection Attributes	22-2
Managing MQSeries Metadata	22-2
Creating Synonyms	22-3
Retrieving Foreign Messages From the Queue	22-4
Data Type Support	22-5
23. Getting Started in Digital Standard MUMPS	23-1
Preparing the MUMPS Environment	23-2
Configuring the Data Adapter for MUMPS	23-2
Declaring Connection Attributes	23-2
Authenticating a User	23-2
Managing MUMPS Metadata	23-3
Creating Synonyms	23-3
Data Type Support	23-6
24. Getting Started in MySQL	24-1
Preparing the MySQL Environment	24-2
Configuring the Data Adapter for MySQL	24-2
Declaring Connection Attributes	24-2
Connecting to an MySQL Database Server	24-2
Authenticating a User	24-3
Adapter Configuration	24-4
Overriding the Default Connection	24-4
Controlling Connection Scope	24-6
Managing MySQL Metadata	24-6
Creating Synonyms	24-6
Data Type Support	24-11
Precision and Scale	24-13
CLOB Activation	24-15
Customizing the MySQL Adapter Environment	24-16
PASSRECS	24-16

25. Getting Started in Nucleus	25-1
Preparing the Nucleus Environment	25-2
Configuring the Data Adapter for Nucleus	25-3
Declaring Connection Attributes	25-3
Overriding the Default Connection	25-5
Controlling the Connection Scope	25-5
Managing Nucleus Metadata	25-6
Creating Synonyms	25-6
Data Type Support	25-12
Changing the Precision and Scale of Numeric Columns	25-13
Customizing the Nucleus Environment	25-14
Obtaining the Number of Rows Updated or Deleted	25-15
26. Getting Started in ODBC	26-1
Preparing the ODBC Environment	26-2
Configuring the Data Adapter for ODBC	26-2
Declaring Connection Attributes	26-2
Overriding the Default Connection	26-3
Controlling the Connection Scope	26-4
Managing ODBC Metadata	26-5
Identifying the Data Adapter	26-5
Accessing Database Tables	26-5
Creating Synonyms	26-5
Data Type Support	26-11
Changing the Precision and Scale of Numeric Columns	26-11
Customizing the ODBC Environment	26-13
Specifying a Timeout Limit	26-13
27. Getting Started in Oracle	27-1
Preparing the Oracle Environment	27-2
Connecting to a Remote Oracle Database Server	27-4
XA Support	27-4
Configuring the Data Adapter for Oracle	27-5
Declaring Connection Attributes	27-5
Connection Syntax in Earlier Server Releases	27-8
Overriding the Default Connection	27-9
Controlling the Connection Scope	27-10

Managing Oracle Metadata	27-10
Creating Synonyms	27-10
Accessing Multiple Database Servers in One SQL Request	27-16
Data Type Support	27-17
Controlling the Mapping of Large Character Data Types	27-19
Controlling the Mapping of Variable-Length Data Types	27-20
Considerations for the CHAR and VARCHAR2 Data Types	27-21
Changing the Precision and Scale of Numeric Columns	27-22
Considerations for the NUMBER Data Type	27-24
Customizing the Oracle Environment	27-25
Designating a Default Tablespace	27-25
Specifying the Block Size for Array Retrieval	27-26
Overriding Default Parameters for Index Space	27-27
Activating NONBLOCK Mode	27-28
Obtaining the Number of Rows Updated or Deleted	27-29
Specifying the Maximum Number of Parameters for Stored Procedures	27-29
Specifying a Timeout Limit	27-30
Calling an Oracle Stored Procedure Using SQL Passthru	27-30
28. Getting Started in PeopleSoft	28-1
Preparing to Configure the Data Adapter for PeopleSoft	28-2
Security	28-2
Location of the Data	28-3
Configuration Worksheet	28-4
Preparing the Server Environment for Adapter Configuration	28-5
Configuring the Server	28-6
Modifying Environment Settings for PeopleSoft RDBMS Connectivity	28-6
Configuring for PeopleSoft 8 Password Authentication	28-9
Starting the Server	28-11
Modifying the Server Profile	28-12
Adding the PeopleSoft Adapter Paths	28-12
Enabling Integrated PeopleSoft Security	28-13
Additional Global Settings for PeopleSoft	28-14
Adding the RDBMS Adapter	28-15
Configuring the Data Adapter for PeopleSoft	28-15
Managing Multiple Connections to PeopleSoft Data Sources	28-16
Adding the First PeopleSoft Connection	28-17
Adding Additional PeopleSoft Connections	28-22
Removing a PeopleSoft Connection	28-23
Updating a PeopleSoft Connection	28-24
Using the PeopleSoft Metadata and Security Administrator	28-25
Accessing the PeopleSoft Metadata and Security Administrator	28-26

Managing PeopleSoft Metadata	28-27
Creating Synonyms	28-27
Removing Synonyms	28-34
Resynchronizing Synonyms	28-35
Security Access Administration	28-36
Adding Security Access	28-36
Removing Security Access	28-39
Resynchronizing Security	28-40
Administration Reports	28-41
Advanced Topics	28-42
Stand-alone Server Usage	28-42
Batch Mode Security Resynchronization	28-45
Troubleshooting Tips	28-48
29. Getting Started in Progress	29-1
Preparing the Progress Environment	29-2
Configuring the Data Adapter for Progress	29-3
Connecting to a Progress Database Server	29-3
Declaring Connection Attributes	29-3
Overriding the Default Connection	29-6
Controlling the Connection Scope	29-7
Managing Progress Metadata	29-7
Creating Synonyms	29-7
Data Type Support	29-12
Controlling the Mapping of Large Character Data Types	29-13
Controlling the Mapping of Variable-Length Data Types	29-14
Changing the Precision and Scale of Numeric Columns	29-16
Customizing the Progress Environment	29-17
Specifying the Block Size for Array Retrieval	29-18
Activating NONBLOCK Mode	29-19
Obtaining the Number of Rows Updated or Deleted	29-19
Specifying a Timeout Limit	29-20
30. Getting Started in Rdb	30-1
Preparing the Rdb Environment	30-2
Configuring the Data Adapter for Rdb	30-2
Overriding the Default Connection	30-4
Controlling the Connection Scope	30-5
Managing Rdb Metadata	30-5
Creating Synonyms	30-5
Data Type Support	30-11
Joining Rdb Tables in Separate Rdb Databases	30-12
Rdb Database Driver Performance	30-13

31. Getting Started in Red Brick	31-1
Preparing the Red Brick Environment	31-2
Configuring the Data Adapter for Red Brick	31-2
Declaring Connection Attributes	31-2
Overriding the Default Connection	31-5
Controlling the Connection Scope	31-6
Managing Red Brick Metadata	31-6
Creating Synonyms	31-6
Data Type Support	31-12
Controlling the Mapping of Variable-Length Data Types	31-13
Changing the Precision and Scale of Numeric Columns	31-13
Customizing the Red Brick Environment	31-15
Specifying a Timeout Limit	31-15
32. Getting Started in RMS	32-1
Manually Describing RMS Files	32-2
File Attributes	32-2
Segment Attributes	32-3
Field Attributes	32-4
Describing Indexed Files (SUFFIX=RMS)	32-12
Segment Name for Indexed Files	32-12
Describing Keys	32-12
Single-Field Secondary Keys	32-16
Describing Complex RMS Keyed	32-17
Describing Multiple Record Types	32-17
Using RECTYPE	32-17
Describing Related Record Types	32-20
Describing Unrelated Record Types	32-21
Describing Embedded Repeating Data	32-23
Assigning a Keyed RMS File to a Master File	32-33
File and Record Locking	32-34
File Locking and the Interface to RMS	32-34
Record Locking	32-35
Handling Locked Records During Table Read Request	32-35
Access File Examples	32-37
Retrieving Data From RMS Files	32-37
Index Selection	32-37
Automatic Index Selection (AIS)	32-38
Requirements for Using AIS	32-38
Syntax for RMS Master File Attributes	32-39
File Attributes	32-39
Segment Attributes	32-40
Field Attributes	32-41

RMS Attribute Summary	32-42
Read/Write Usage Limitations of the Data Adapter for RMS	32-56
OCCURS Statements in a Master File	32-56
Commit Processing	32-56
SQL Commands	32-57
SQL, MODIFY and MAINTAIN Operations	32-57
33. Getting Started in SAP Business Intelligence Warehouse (BW)	33-1
Preparing the SAP BW Environment	33-2
Accessing Multiple Systems	33-4
Configuring the Query Adapter for SAP BW	33-5
Declaring Connection Attributes	33-5
Extracting Data With SAP BW Queries	33-8
Defining New Business Explorer Queries	33-9
Restricting Query Characteristics	33-10
Restricting and Calculating Key Figures	33-11
Viewing Query Properties and Releasing for OLAP	33-12
Managing SAP BW Metadata	33-13
Understanding Field Names	33-13
Creating Synonyms	33-14
Mapping Metadata	33-17
Variable Types	33-21
Overview of SAP BW Reporting Concepts	33-22
Reporting Rules	33-23
General Tips for Reporting	33-24
Using Columnar Reports to Slice Cubes	33-24
Reporting With Variables	33-25
Working With Hierarchies	33-27
Creating Dynamic Dimensions	33-27
BW Explorer	33-28
Using BW Explorer	33-30
Producing SAP BW Requests	33-31
34. Getting Started in SAP/R3	34-1
Preparing the SAP R/3 Environment	34-2
Accessing Multiple SAP R/3 Systems	34-11
Configuring the Data Adapter for SAP R/3	34-12
Declaring Connection Attributes	34-12
Managing SAP R/3 Metadata	34-18
Creating Synonyms	34-18
SAP R/3 Table Support	34-30
SAP R/3 Data Type Support	34-31
SAP R/3 Open/SQL Support	34-32

Advanced SAP R/3 Features	34-33
Support for User Security	34-33
BAPI Support	34-33
Joins Support	34-34
Setting up the Report Processing Mode	34-34
Dialog Process	34-34
Batch Mode Processing	34-35
Supporting Mixed Code Page Environments	34-35
Character Conversion Tables	34-35
Tracing Options	34-41
Producing SAP R/3 Requests	34-41
35. Getting Started in Siebel	35-1
Software Requirements	35-2
Preparing the Siebel Environment	35-2
Setting Security	35-2
Accessing a Server	35-2
Preparing the Server Environment for Adapter Configuration	35-3
Defining Environment Variables	35-3
Adding a JSCOM3 Listener	35-4
Configuring the Data Adapter for Siebel	35-6
Declaring Connection Attributes	35-6
Overriding the Default Connection	35-9
Managing Siebel Metadata	35-9
Creating Synonyms	35-10
Data Type Support	35-17
36. Getting Started in Supra	36-1
Preparing the Supra Environment	36-2
Configuring the Data Adapter for Supra	36-8
Declaring Connection Attributes	36-8
Supra Overview and Mapping Considerations	36-9
Mapping Concepts	36-9
Supra RDM	36-9
Mapping of Supra Views and Fields	36-11
Managing Supra Metadata	36-11
Master File	36-11
Access File	36-13
Embedded Cross-Reference	36-15
Supra PDM	36-15

Supra Modules	36-15
Module CFDP4001	36-15
Module CFDP4002	36-16
Module CFDP4003	36-16
Adapter Tracing	36-17
37. Getting Started in Sybase ASE	37-1
Preparing the Sybase ASE Environment	37-2
Identifying the Location of the <i>Interfaces</i> File	37-2
Specifying the Sybase Server Name	37-2
Accessing a Remote Sybase Server	37-3
XA Support	37-3
Configuring the Data Adapter for Sybase ASE	37-3
Declaring Connection Attributes	37-4
Overriding the Default Connection	37-6
Controlling the Connection Scope	37-7
Managing Sybase ASE Metadata	37-7
Creating Synonyms	37-7
Accessing Different Databases on the Same Sybase Server	37-12
Data Type Support	37-13
Controlling the Mapping of Large Character Data Types	37-15
Controlling the Mapping of Variable-Length Data Types	37-16
Changing the Precision and Scale of Numeric Columns	37-17
Customizing the Sybase ASE Environment	37-19
Specifying the Block Size for Array Retrieval	37-19
Activating NONBLOCK Mode	37-20
Obtaining the Number of Rows Updated or Deleted	37-21
Calling a Sybase ASE Stored Procedure Using SQL Passthru	37-22
38. Getting Started in Sybase IQ	38-1
Preparing the Sybase IQ Environment	38-2
Identifying the Location of the <i>Interfaces</i> File	38-2
Specifying the Sybase Server Name	38-3
Accessing a Remote Sybase Server	38-3
Configuring the Data Adapter for Sybase IQ	38-3
Declaring Connection Attributes	38-3
Overriding the Default Connection	38-5
Controlling the Connection Scope	38-6

Managing Sybase IQ Metadata	38-7
Creating Synonyms	38-7
Accessing Different Databases on the Same Sybase Server	38-12
Data Type Support	38-13
Controlling the Mapping of Large Character Data Types	38-14
Controlling the Mapping of Variable-Length Data Types	38-15
Changing the Precision and Scale of Numeric Columns	38-16
Customizing the Sybase IQ Environment	38-19
Activating NONBLOCK Mode	38-19
Obtaining the Number of Rows Updated or Deleted	38-20
Calling a Sybase IQ Stored Procedure Using SQL Passthru	38-21
39. Getting Started in Teradata	39-1
Preparing the Teradata Environment	39-2
Configuring the Data Adapter for Teradata	39-3
Declaring Connection Attributes	39-3
Overriding the Default Connection	39-5
Controlling the Connection Scope	39-6
Managing Teradata Metadata	39-6
Creating Synonyms	39-6
Data Type Support	39-12
Controlling the Mapping of Large Character Data Types	39-14
Controlling the Mapping of Variable-Length Data Types	39-15
Changing the Precision and Scale of Numeric Columns	39-16
Customizing the Teradata Environment	39-17
Controlling the Column Heading in a Request	39-18
Obtaining the Number of Rows Updated or Deleted	39-18
40. Getting Started in UniVerse	40-1
Preparing the UniVerse Environment	40-2
Configuring the Data Adapter for UniVerse	40-2
Declaring Connection Attributes	40-2
Overriding the Default Connection	40-4
Controlling the Connection Scope	40-5
Managing UniVerse Metadata	40-5
Creating Synonyms	40-5
Data Type Support	40-11
Changing the Precision and Scale of Numeric Columns	40-12
Customizing the UniVerse Environment	40-13
Displaying Multivalued Columns in UniVerse	40-14
Obtaining the Number of Rows Updated or Deleted	40-14
Controlling Transactions	40-15

41. Getting Started in XML	41-1
Preparing the XML Environment	41-2
Configuring the Data Adapter for XML	41-2
Associating a Master File With an XML Document	41-2
Accessing XML Documents From Relational DBMS CLOB Fields	41-3
Managing XML Metadata	41-4
Creating Synonyms	41-4
Data Type Support	41-9
Conversion	41-10
Numeric Values	41-10
Dates in XML	41-10
Using XML XFOCUS	41-11
Using XML XFOCUS to Execute XML Documents	41-11
Using XML XFOCUS to Archive XML Documents	41-12
A. XA Support	A-1
XA Transaction Management	A-2
Supported Interfaces	A-3
Implementation	A-3
Vendor Specifics	A-4

CHAPTER 1

Introduction to iWay Data Adapters

Topics:

- Functions of a Data Adapter
- Data Management
- Metadata Services With SQLENGINE SET
- Additional Master File Attributes

In client/server architecture, data adapters enable clients to manage data from virtually any data source, and on any operating system, through the use of SQL statements.

The client generates requests for data residing on the server. The server acts as a source of data, and can accept requests from multiple clients for data access and manipulation. Client/server architecture divides a traditional single system into a front-end and a back-end. The workload is distributed between the client and the server. Communications software establishes the link between client and server, and interfaces to the desired communications protocol.

A client application uses SQL as the standard access language for requests to all relational and non-relational data. Depending on the environment—that is, the hardware and software employed throughout the enterprise—an applicable communications protocol transmits the request for data to the server, and then returns the answer set to the client.

The server in client/server architecture is used for data access and manipulation. The server receives a request for data, processes it, and returns an answer set or message to the client. Data adapters are among the various subsystems that comprise a server.

Functions of a Data Adapter

The server uses data adapters to access data sources. The server receives SQL requests from the client and passes them to the data adapter in a standard format. The data adapter takes the request, transforms it into the native data manipulation language (DML), and then issues calls to the data source using its API. In this way, the data adapter insulates the server from details of the data source. An application can issue SQL statements or call stored procedures.

Data adapters are available for many data sources. Every data adapter is specifically designed for the data source that it accesses, and, as a result, is able to translate between SQL and the DML of the data source. Data adapters provide solutions to product variations, including product differences in syntax, functionality, schema, data types, catalogs, data representations, message processing, and answer set retrieval.

How a Data Adapter Works

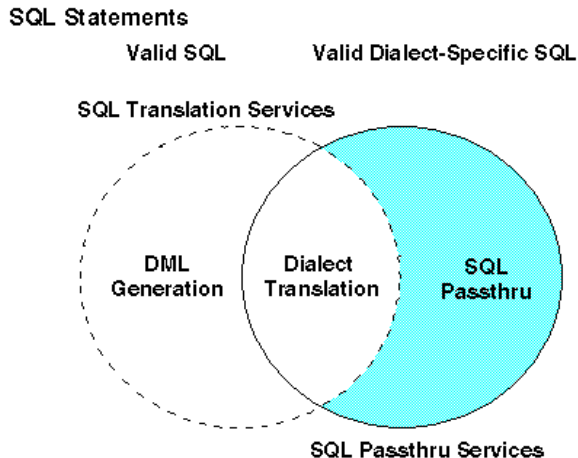
The data adapter manages the communication between the data interface and the data source, passing data management requests to the data source and returning either answer sets or messages to the requestor.

To perform these functions, the data adapter:

1. Translates the request to the applicable DML.
2. Attaches to the targeted data source, using standard attachment calls. The data adapter then passes the request to the data source.
3. The data source processes the request.
4. The results or error conditions are returned to the client application for further processing.

Processing SQL Requests

A server can be configured to behave in different ways upon receipt of SQL requests from a client application. A server handles SQL requests for data in the following ways, as illustrated in the diagram:



- **Direct Passthru.** When Direct Passthru is enabled, the server passes SQL requests directly to the specified RDBMS for processing. The name of the targeted RDBMS (the database engine) is supplied in the server profile or, in some cases, by the client application. A Full-Function or Hub Server can operate temporarily in Direct Passthru mode when invoked by a client application. The user is responsible for activating and deactivating Direct Passthru as needed. Direct Passthru is described in more detail in the *SQL Reference* manual.
- **SQL Processing.** When the database engine is not set in the server profile or supplied by a client application, Direct Passthru is not enabled. Instead, a Hub or Full-Function Server invokes its default behavior: it accepts the incoming SQL request and verifies that it is valid. Then the server determines if it can process the incoming SQL request:
 - If the request meets certain requirements, the server passes it directly to the RDBMS for processing. This is called Automatic Passthru. Automatic Passthru is described in more detail in the *SQL Reference* manual.
 - If the syntax of the request does not conform to the syntax of the RDBMS, the server translates the request into internal DML and passes it to the data adapter. The data adapter generates adapter-specific DML and passes the request to the RDBMS for processing. This is called SQL translation. SQL translation is described in more detail in the *SQL Reference* manual.

Relational and Non-Relational Data Adapters

Data adapters can retrieve answer sets from both relational and non-relational data sources. Since the architectures of relational and non-relational data structures vary, the relational and non-relational data adapters adjust for these differences. For example, relational data adapters are designed to handle data sources that contain data in rows and columns in tables, while non-relational data adapters are designed to accommodate the architecture of each distinct data source, for instance, a hierarchical or network data source, or a sequential or indexed file system.

The following table lists key features of relational and non-relational data adapters:

Feature	Relational Data Adapters	Non-Relational Data Adapters
DEFINE (virtual field) in Master File	Yes	Yes
Access Control	Yes	Yes
Transaction Management Commands	Yes	No

Relational and non-relational data adapters:

- Allow the data source to perform the work required to join, sort, and aggregate data. Therefore, the volume of data source-to-server communication is reduced, resulting in better response times for users. The data adapter tries to optimize the queries as best it can.
- Communicate with the data source through DML statements. You can view these DML statements with traces provided by the trace facilities. These traces are helpful for debugging your procedure or for data adapter performance analysis. Note that traces:
 - Are common for all relational data adapters.
 - Vary for non-relational data adapters to accommodate the differences in data structures and DML calls.
- Support both DEFINE (virtual) fields and DBA.
- Relational data adapters include a variety of COMMIT, connection, and thread control commands that enable Database Administrators to control the opening and closing of connections and choose when to commit or rollback transactions. This level of transaction control is not supported by non-relational data adapters. In Full-Function Server mode, COMMIT/ROLLBACK will be propagated to all relational data sources local to the server. If the hub is active, a COMMIT will be issued against remote data servers as well. All PREPARE handles are cleared (this is a Broadcast COMMIT).

To address these similarities and differences, the *Server Administration* manual contains:

- *General* components with information that is common to all data adapters, as well as information that is common either to all relational or to all non-relational data adapters.
- *Customized* components with information that applies to specific data adapters.

Supported Data Sources

The following table lists supported data sources:

IBM-Compatible Mainframes (MVS/VM)	OpenVMS	UNIX-Based Computers
ADABAS CA-Datcom/DB DB2 FOCUS CA-IDMS/DB CA-IDMS/SQL IMS/DB ISAM Millennium MODEL 204 NOMAD Oracle SQL/DS Supra System 2000 Teradata TOTAL QSAM VSAM	ADABAS/C DBMS FOCUS Ingres Oracle Rdb RMS Sybase Progress	ADABAS/C C-ISAM DB2/6000 Essbase FOCUS Interplex (DMS/RDMS 2200/1100) Informix Ingres Oracle Progress Red Brick Sybase ASE Sybase IQ Teradata UniVerse (PICK)

OS/400	Tandem	Windows
FOCUS SQL/400	Enscribe FOCUS NonStop SQL	ADABAS/C DB2/2 Essbase FOCUS Informix Interplex (DMS/RDMS 2200/1100) Oracle Microsoft SQL Server Microsoft Analytical Engine Sybase ASE Sybase IQ Teradata

Data Management

As you manage your data, you may be required to modify your server and communications configuration files. The first step is understanding how and where data is described and the roles of the server and data adapters in managing the processing flow.

Describing Data Sources

In order to access a table or view, you must first describe it using two files: a Master File and an associated Access File.

Master Files and Access Files can represent an entire table or part of a table. Also, several pairs of Master and Access Files can define different subsets of columns for the same table, or one pair of Master and Access Files can describe several tables.

Note: In these topics, the term table refers to both base tables and views in data sources. The Master File describes the columns of the data source table using keywords in comma-delimited format. The Access File includes additional parameters that complete the definition of the data source table. Some data adapters require both files to fulfill queries, and to build the DML to access the non-SQL data sources.

Processing Requests

When requests are processed, control is passed from the server to a data adapter and back. During the process, selected information is read from the Master and Access Files as described below.

The server processes a request as follows:

1. The request is parsed to identify the table.
2. The Master File for the table is read.
3. The SUFFIX value in the Master File is checked (SUFFIX indicates the type of data source).
4. Control is passed to the appropriate data adapter.

The data adapter then:

5. Locates the corresponding Access File.
6. Uses the information contained in the Master and Access Files to generate the DML statements (if necessary) required to accomplish the request.
7. Passes the DML statements to the data source.
8. Retrieves the answer set generated by the data source.
9. Returns control to the server.

Depending on the requirements of the request, additional processing may be performed on the returned data.

Master File

A Master File describes a logical data source. A logical data source can be made up of one or more physical data sources of the same type. Each segment is a physical data source.

Master Files contain three types of declarations:

Declaration Type	Description
File	Names the file and describes the type of data source.
Segment	Identifies a table, file, view, or segment.
Field	Describes the columns of the table, view, or fields in the file.

The following guidelines apply:

- A declaration consists of attribute-value pairs separated by commas.
- Each declaration must begin on a separate line. A declaration can span as many lines as necessary, as long as no single keyword-value pair spans two lines.
- Do not use system or reserved words as names for files, segments, fields, or aliases. Specifying a reserved word generates syntax errors.

Syntax

How to Specify a File Declaration in a Master File

A Master File begins with a file declaration, which has at least two attributes:

`FILENAME (FILE)`

Identifies the Master File.

`SUFFIX`

Identifies the data adapter needed to interpret the request.

The syntax for a file declaration is

`FILE[NAME]=file, SUFFIX=suffix [,,$]`

where:

file

Is the file name for the Master File. The file name should start with a letter and be representative of the table or view contents. The actual file must have a .mas extension, but the value for this attribute should not include the extension. The file name without the .mas extension can consist of a maximum of eight alphanumeric characters.

suffix

Identifies the data adapter needed to interpret the request. For example, SQLORA is the value for the Oracle Data Adapter.

Syntax How to Specify a Segment Declaration in a Master File

Each table described in a Master File requires a segment declaration. The segment declaration consists of at least two attributes:

SEGNAME

Identifies one table.

SEGTYPE

Identifies the physical storage of rows and the uniqueness of column values.

The syntax for a segment declaration is

```
SEGNAME=segname , SEGTYPE=S0 [ , $ ]
```

where:

segname

Is the segment name which serves as a link to the actual table name. It may be the same as the name chosen for FILENAME, the actual table name, or an arbitrary name. It can consist of a maximum of 8 alphanumeric characters.

The SEGNAME value in the Master File must be the same as the SEGNAME value specified in the Access File, where the TABLENAME portion of the segment declaration contains the fully-qualified name of the table.

S0

Indicates that the RDBMS is responsible for both physical storage of rows and the uniqueness of column values (if a unique index or constraint exists). It always has a value of S0 (S zero).

Syntax How to Specify a Field Declaration in a Master File

Each row in a table may consist of one or more columns. These columns are described in the Master File as fields with the following primary field attributes:

FIELDNAME

Identifies the name of a field.

ALIAS

Identifies the full column name.

USAGE

Identifies how to display a field on reports.

ACTUAL

Identifies the data type and length in bytes for a field.

MISSING

Identifies whether a field supports null data.

You can obtain values for these attributes by using the system catalog table ALL_TAB_COLUMNS for the existing table or view you wish to describe.

The syntax for a field declaration is

```
FIELD[NAME]=fieldname, [ALIAS=]sqlcolumn, [USAGE=]display_format,  
[ACTUAL=]storage_format [,MISSING={ON|OFF}], $
```

where:

fieldname

Is the name of the field. This value must be unique within the Master File. The name can consist of a maximum of 48 alphanumeric characters including letters, digits, and underscores. The name must begin with a letter. Special characters and embedded blanks are not recommended. The order of field declarations in the Master File is significant with regard to the specification of key columns. For more information, see *Primary Key* on page 1-13.

Tip: Since the name appears as the default column title for reports, for client applications, or EDADESCRIBE, select a name that is representative of the data.

It is not necessary to describe all the columns of the table in your Master File.

sqlcolumn

Is the full column name (the data adapter uses it to generate SQL statements). This value must comply with the naming conventions for identifiers, where a name should start with a letter and may be followed by any combination of letters, digits, or underscores. Embedded spaces are not allowed.

display_format

Is the display format. The value must include the field type and length and may contain edit options.

The data type of the display format must be identical to that of the ACTUAL format. For example, a field with an alphanumeric USAGE data type must have an alphanumeric ACTUAL data type.

Fields or columns with decimal or floating point data types must be described with the correct scale (s) and precision (p). Scale is the number of positions to the right of the decimal point. Precision is the total length of the field.

For the server, the total display length of the field or column *includes* the decimal point and negative sign. In SQL, the total length of the field or column *excludes* the decimal point and negative sign. For example, a column defined as DECIMAL(5,2) would have a USAGE attribute of P7.2 to allow for the decimal point and a possible negative sign.

storage_format

Is the storage format of the data type and length in bytes. For more information on data type support, see the *SQL Reference* manual.

ON

Displays the character specified by the NODATA parameter for missing data. For related information, see *MISSING Attribute* on page 1-11.

OFF

Displays blanks or zeroes for fields having no value. This is the default. See *MISSING Attribute* on page 1-11.

MISSING Attribute

In a table, a null value represents a missing or unknown value; it is not the same as a blank or a zero. For example, a column specification that allows null values is used where a column need not have a value in every row (such as a raise amount in a table containing payroll data).

- The default NODATA character is a period.
- A column in a table that allows null data does not need to include the NULL clause in its table definition, since that is the default.
- In the Master File for that table, the column that allows null data must be described with the MISSING attribute value ON. The default for this attribute is OFF, which corresponds to the NOT NULL attribute in the table definition.

If the column allows null data but the corresponding field in the Master File is described with the MISSING attribute value OFF, null data values appear as zeroes or blanks.

Access File

Each Master File may have a corresponding Access File. The name of the Access File must be identical to that of the Master File, but the extension will be .acx instead of .mas.

The Access File serves as a link between the server and the data source by providing the means to associate a segment in the Master File with the table it describes. The Access File minimally identifies the table and primary key (if there is one). It may also indicate the logical sort order of data and identify storage areas for the table.

Syntax **How to Specify a Segment Declaration in an Access File**

The segment declaration in the Access File establishes the link between one segment of the Master File and the actual table or view. Attributes that constitute the segment declaration are:

SEGNAME

Identifies one table.

TABLENAME

Identifies the table or view. It may contain the owner ID as well as the table name.

KEYS

Identifies how many columns constitute the primary key.

KEYORDER

Identifies the logical sort sequence of data by the primary key.

The syntax for a segment declaration in an Access File is

```
SEGNAME=segname, TABLENAME=owner.tablename databaselink  
[ ,KEYS={n|0}] [ ,KEYORDER={LOW|HIGH}] , $
```

where:

segname

Is the same value as the SEGNAME value in the Master File.

owner

Is the user ID by default.

tablename

Is the name of the table or view.

databaselink

Is the DATABASE LINK name to be used in the currently connected database server.

n

Is the number of columns that constitute the primary key. It can be a value from 0 to 16. The default value is 0. For more information, see *Primary Key* on page 1-13.

LOW

Indicates an ascending primary key logical sort order. This value is the default.

HIGH

Indicates a descending primary key logical sort order.

Primary Key

A table's primary key consists of the column or combination of columns whose values uniquely identify each row of the table. In the employee table, for example, every employee is assigned a unique employee identification number. Each employee is represented by one and only one row of the table, and is uniquely identified by that identification number.

The primary key definition must be defined partly in the Master File and partly in the Access File:

- The order of field declarations in the Master File is significant to the specification of key columns. To define the primary key in a Master File, describe its component fields immediately after the segment declaration. You can specify the remaining fields in any order. In the Access File, the KEYS attribute completes the process of defining the primary key.
- To identify the primary key, the data adapter uses the number of columns (*n*) indicated by the KEYS attribute in the Access File and the first *n* fields described in the Master File.

Typically, the primary key is supported by the creation of a unique index in the SQL language to prevent the insertion of duplicate key values. The data adapter itself does not require any index on the column(s) comprising the primary key (although a unique index is certainly desirable for both data integrity and performance reasons).

Creating Virtual Fields

You use the DEFINE command to accomplish these tasks.

Syntax How to Create Virtual Fields With the DEFINE Command

```
DEFINE fieldname/format [WITH fieldname]=expression ;$
```

where:

fieldname

Is a field name for the virtual field. It can consist of 1 to 48 characters. You must not qualify the field name.

format

Provides the display format for the field and follows the rules for USAGE formats. This operand is optional. If not specified, the default value is D12.2.

WITH *fieldname*

Must be coded when the expression is a constant. Any real field can be chosen from the same segment the DEFINE is associated with.

expression

Can be either a mathematical or a logical statement. It can consist of constants, database fields, and virtual fields. The expression must end with a semicolon followed by a dollar sign (;\$).

Place your DEFINE statements after all of the field descriptions in the segment. If you are using the DESCRIPTION or TITLE attributes with virtual fields, you must place these attributes on a separate line.

Example Defining a Virtual Field in a Master File

In the example that follows, the virtual field PROFIT is defined at the end of the segment named BODY.

```
SEGMENT=BODY, SEGTYPE=S0 , PARENT=CARREC,$
  FIELDNAME=BODYTYPE           ,ALIAS=BODYTYPE           ,A12,A12,$
FIELDNAME=DEALER_COST         ,ALIAS=DEALER_COST     ,D8, D8 ,,$
FIELDNAME=RETAIL_COST         ,ALIAS=RETAIL_COST     ,D8, D8 ,,$
DEFINE PROFIT/D8 = RETAIL_COST - DEALER_COST
  ;DESC=NET_COST, TITLE='NET,COST' ,,$
```

As a result of this DEFINE statement, you can use PROFIT as a field name in reports. PROFIT is treated as a field with a value equal to the value of RETAIL_COST minus DEALER_COST.

Note:

- Since the complete data source needs to be read to calculate virtual fields, screening conditions on virtual fields may incur additional overhead.
- Virtual fields in the Master File for relational and remote data sources will, if referenced in a query, disable Automatic Passthru.

Cross-Century Dates

Many existing business applications use two digits to designate a year, instead of four digits. When they receive a value for a year, such as 00, they typically interpret it as 1900, assuming that the first two digits are 19, for the twentieth century. There is considerable risk that date-sensitive calculations in existing applications will be wrong unless an apparatus is provided for determining the century in question. This will impact almost every type of application, including those that process mortgages, insurance policies, anniversaries, bonds, inventory replenishment, contracts, leases, pensions, receivables, and customer records.

The cross-century dates feature enables you to solve this problem at the file and field level of your applications. You can retain your global settings while changing the file-level settings for greater flexibility.

You can enable this feature:

- Using SET commands.
- At the file level in a Master File.
- At the field level in a Master File.

Cross-Century Dates SET Commands

The server delivers SET commands that provide a means of interpreting the century if the first two digits of the year are not provided:

```
SET DEFCENT
SET YRTHRESH
```

If the first two digits are provided, they are simply accepted and validated.

Syntax How to Implement a Cross-Century Date

The DEFCENT syntax is

```
SET DEFCENT=nn
```

where:

nn

Is 19 unless otherwise specified.

The YRTHRESH syntax is

```
SET YRTHRESH=nn
```

where:

nn

Is zero unless otherwise specified.

The combination of DEFCENT and YRTHRESH establishes a base year for a 100-year window. Any 2-digit year is assumed to fall within that window, and the first two digits are set accordingly. Years outside the declared window must be handled by user coding.

The default values for the two commands are SET DEFCENT=19, SET YRTHRESH=00. When you provide a year threshold, years greater than or equal to that value assume the value assigned by DEFCENT. Years lower than that threshold become DEFCENT plus 1.

To see how DEFCENT and YRTHRESH are applied to interpret 2-digit years, consider the following:

```
SET DEFCEM=19, SET YRTHRESH=80
```

This set of commands describes a range from 1980 to 2079. If a 2-digit year field contains the value 99, then the server interprets the year as 1999. If the year field is 79, then the year is interpreted as 2079. If the year field is 00, then the year is interpreted as 2000.

Master File Syntax

Instead of using SET commands, you can include settings at the file level in a Master File, or at the field level in a Master File.

Syntax How to Add Cross-Century Date Settings at the File Level

The FDEFCEM syntax is

```
{FDEFCEM | FDFC} =nn
```

where:

nn

Is 19, unless otherwise specified.

The FYRTHRESH syntax is

```
{FYRTHRESH | FYRT} =nn
```

where:

nn

Is zero, unless otherwise specified.

Syntax How to Add Cross-Century Date Settings at the Field Level

At the field level, DEFCEM and YRTHRESH can be added. The DEFCEM syntax is

```
{DEFCEM | DFC} =nn
```

where:

nn

Is 19, unless otherwise specified.

The YRTHRESH syntax is

```
{YRTHRESH | YRT} =nn
```

where:

nn

Is zero, unless otherwise specified.

Syntax **How to Add Cross-Century Dates Using a DEFINE Command**

```
DEFINE FILE EMPLOYEE
  fld/fmt [{DEFCENT|DFC} nm {YRTHRESH|YRT} nm] [MISSING...]=expression;
END
```

The DFC and YRT syntax must follow the field format information.

Example **Implementing Cross-Century Dates**

The following example illustrates how century interpretation is implemented at both the file level and field level in a Master File.

```
FILENAME=EMPLOYEE, SUFFIX=FOC, FDEFCENT=20, FYRTHRESH=66,$
SEGNAME=EMPINFO, SEGTYPE=S1
  FIELDNAME=EMP_ID, ALIAS=EID, FORMAT=A9, $
  FIELDNAME=LAST_NAME, ALIAS=LN, FORMAT=A15, $
  FIELDNAME=FIRST_NAME, ALIAS=FN, FORMAT=A10, $
  FIELDNAME=HIRE_DATE, ALIAS=HDT, FORMAT=I6YMD, DEFCENT=19,
YRTHRESH=75,$
```

The next example illustrates the conversion of a 2-digit year field with the DEFINE command:

```
DEFINE FILE EMPLOYEE
ESHIRE_DATE/YYMD = HIRE_DATE; (The format of HIRE_DATE is I6YM.)
ESHIRE DFC 19 YRT 80 = HIRE_DATE;
END
```

Metadata Services With SQLENGINE SET

When the server is dedicated to accessing one Relational Database Management System (RDBMS) using the SET SQLENGINE command in the global profile, metadata calls to the server are processed against the native catalogs of the RDBMS.

How Applications Access Metadata

The metadata procedures used by applications to query the native catalog directly are:

- For an ODBC-enabled application, ODBC SQL calls.
- For an API-enabled application, EDARPC for ODBCxxxx calls.

The following table shows the relationship between the ODBC call and the API call:

ODBC Call	API Call
SQLTables	ODBCTABL
SQLColumns	ODBCCOLS

SQLPrimaryKeys	ODBCPKEY
SQLStatistics	ODBCSTAT
SQLProcedures	ODBCPROC
SQLProcedureColumns	ODBCPRCC
SQLSpecialColumns	ODBCSCOL
SQLColumnPrivileges	ODBCPRV
SQLForeignKeys	ODBCFKEY
SQLTablePrivileges	ODBCTPRV

You can use the following two commands to control or override table and column metadata calls:

```
SQL sqlengine SET ODBCCOLSSORT
```

```
SQL sqlengine SET ODBCTABL
```

You can include these commands in any of the supported server profiles.

Obtaining Column Information (DB2 only)

When you issue either the SQLColumns or ODBCCOLS Metadata call from a client application, by default, the server requests the columns from DB2 in *colno* order.

Syntax How to Request the Sort Order of Column Data

```
SET ODBCCOLSSORT {ON|OFF}
```

where:

ON

The column data is requested from the server with order by colno. This value is the default.

OFF

The column data is returned in unsorted order.

Obtaining a User-Defined Metadata

When a client application issues either an ODBC or API metadata call, the server will run internal procedures that issue default SQL against the native RDBMS catalogs. You can issue your own SQL to run in place of the default SQL. You can specify this type of override for any of the internal server routines that deal with metadata.

Syntax **How to Request User-Defined Metadata**

SQL *sqlengine* SET ODBCxxxx *procname*

where:

ODBCxxxx

Is the internal server routine name. Possible values are: ODBCTABL, ODBCCOLS, ODBCPKEY, ODBCSTAT, ODBCPROC, ODBCPRCC, ODBCSCOL, ODBCCPRV, ODBCFCKEY, and ODBCTPRV.

procname

Is the procedure to run when the server receives the metadata call. This procedure must be available through the FOCEXEC ddname allocation in the server JCL.

Note: If this override procedure is used on a server running under MVS, it will add approximately 600K of storage above the line for each user.

When coding an override procedure, care must be taken to maintain the select list (answer set layout). The select list must conform to the ODBC specification for the return from the SQLTables call. See the *ODBC 2.0 Programmer's Reference and SDK Guide* for the SQLTables specification layout. Also, when using this override procedure, the parameters that are sent with the Metadata call need to be parsed correctly.

The override procedure should take the following format:

1. Dialogue Manager code to parse metadata call parameters.
2. SQL *sqlengine*
SELECT code;
3. TABLE
ON TABLE PCHOLD FORMAT ALPHA
END

Items 1 and 2 above are user coded; item 3 must always be present at the end of the procedure as coded above.

Example **Returning a List of Tables**

The following sample code found in *qualif.EDARPC.DATA(DB2ODBC1)* returns a list of tables that the connected user is authorized to INSERT, UPDATE, DELETE, and SELECT. It is a sample of how to code a DB2 override procedure for ODBCTABL. It is one of several ways to code SQL to return an answer set for the ODBCTABL call. You can code any relevant query as long as the SELECT list is maintained.

In a server profile, issue SQL DB2 SET ODBCTABL DB2ODBC1, and then execute the following RPC request on the server:

```
ODBCTABL , <NULL> , , , , 0 , 0 , *
```

The following example matches the above ODBCTABL call:

```

-*
-* Dialogue Manager code to parse the ODBCTABL parameter list
-*
-DEFAULTS 1=' ',2='% ',3='% '
-IF &2 NE '<NULL>' THEN GOTO LAB1;
-SET &2 = ' ';
-LAB1
-IF &3 NE ' ' THEN GOTO LAB2;
-SET &3 = '% ';
-LAB2
-*
-* SQL SELECT code
-*
SQL DB2
SELECT ' ',T2.CREATOR,T2.NAME,'TABLE',' '
FROM SYSIBM.SYSTABAUTH T1,SYSIBM.SYSTABLES T2
WHERE T1.GRANTEE = USER
AND T1.TTNAME LIKE '&2'
AND T1.TCREATOR LIKE '&3'
AND T2.TYPE = 'T'
AND (T1.DELETEAUTH IN ('G','Y')
OR T1.INSERTAUTH IN ('G','Y')
OR T1.SELECTAUTH IN ('G','Y')
OR T1.UPDATEAUTH IN ('G','Y'))
AND T1.TTNAME = T2.NAME
AND T1.TCREATOR = T2.CREATOR
UNION
SELECT ' ',T2.CREATOR,T2.NAME,'VIEW',' '
FROM SYSIBM.SYSTABAUTH T1,SYSIBM.SYSTABLES T2
WHERE T1.GRANTEE = USER
AND T1.TTNAME LIKE '&1'
AND T1.TCREATOR LIKE '&2'
AND T2.TYPE = 'V'
AND (T1.DELETEAUTH IN ('G','Y')
OR T1.INSERTAUTH IN ('G','Y')
OR T1.SELECTAUTH IN ('G','Y')
OR T1.UPDATEAUTH IN ('G','Y'))
AND T1.TTNAME = T2.NAME
AND T1.TCREATOR = T2.CREATOR
ORDER BY CREATOR,NAME;
-*
-* The following code must always be present
-*
TABLE
ON TABLE PCHOLD FORMAT ALPHA
END
```


Maintaining Upward Compatibility

In Version 4.3.x or earlier, an Extended Catalog (SYSOWNER table) was created at installation time. This provided additional control to the list of tables returned with the SQLTables or ODBCTABL call. In Version 5.1.0 and higher, this table is no longer created. If you need to use this table from a previous version of the server to continue to have control over the list of tables, issue the following in any supported server profile:

```
SQL sqlengine SET OWNERID ownerid
```

where:

sqlengine

Indicates the data source. You can omit this value if you previously issued the SET SQLENGINE command.

ownerid

Identifies the creator or owner of the Extended Catalog table SYSOWNER. If this command is used, the SQL generated to provide a list of tables will be limited by the owner names in the SYSOWNER table.

This command is only supported for customers who have configured a Relational Gateway in previous releases of the server and want to continue with the same configuration.

Additional Master File Attributes

These topics describe how to modify your server and communications configuration files.

The following attributes enable you to provide descriptive information about tables and columns.

REMARKS

Is an optional attribute for documenting tables.

DESCRIPTION

Is an optional attribute for documenting columns.

TITLE

Is an optional attribute for supplying an alternate column title on a report to replace the FIELDNAME value, which is normally used.

For the server, client tools using the API or ODBC will not use the TITLE attribute. The TITLE attribute is available only when directly querying the server catalog.

Documenting a Table

The REMARKS attribute enables you to document a table.

Syntax How to Document a Table

The syntax is

```
REMARKS=text , $
```

where:

text

Is one line of text. It can consist of a maximum of 78 characters. If the text contains a comma, you must enclose the text in single quotation marks.

The REMARKS attribute cannot span more than one line in the Master File. If necessary, move the entire attribute to a line by itself.

Example Documenting an Oracle Table

The following example shows how to document the Oracle table SAMPLE.

```
FILE=SAMPLE,SUFFIX=SQLORA,REMARKS=This is a sample Oracle table. , $
```

Documenting a Column

The DESCRIPTION attribute enables you to provide a comment for a column in a table.

You can also add documentation to a column declaration—or a segment or table declaration—by typing a comment following the terminating dollar sign. You can even create an entire comment line by inserting a new line following a declaration and placing a dollar sign at the beginning of the line. For the server, a comment added in this manner will not be available to any client application.

If you are using DEFINE fields in a Master File, you must place the DESCRIPTION attribute on a line by itself. The semicolon after the DEFINE must appear on the same line as DESCRIPTION=.

Syntax **How to Document a Column**

The syntax is

```
DESC[RIPTION]=text , $
```

where:

text

Is one line of text. It can consist of a maximum of 44 characters.

If the text contains a comma, you must enclose the text in single quotation marks.

The DESCRIPTION attribute cannot span more than one line in the Master File. If necessary, move the entire attribute to a line by itself.

Note: Whenever possible, place the description on the same line with the attributes FIELDNAME and ALIAS, to conserve space.

Example **Documenting a Column**

The following example shows how to provide a description for the column UNITS. The single quotation marks are required because the description contains a comma.

```
FIELD=UNITS,ALIAS=QTY,FORMAT=I6,DESC='This is quantity sold, not returned', $
```

Example **Documenting a DEFINE Field**

The following example shows how to provide a description for the DEFINE field ITEMS_SOLD.

```
DEFINE ITEMS_SOLD/D8=ORDERED-INVENTORY  
;DESC=DAMAGED ITEMS NOT INCLUDED, $
```

Supplying an Alternate Column Title

When you generate a report, each column title in the report defaults to the name of the column as it appears in the table. However, you can change the default column title by specifying the TITLE attribute.

For the server, client tools using the API or ODBC will not use the TITLE attribute. The TITLE attribute is available only when directly querying the server catalog, or when using WebFOCUS (Windows version).

You can override both the FIELDNAME and TITLE attributes with AS phrases in your report request. To override an existing TITLE attribute, use the SET TITLE command. To control whether the TITLE attribute is propagated in the Master File of a HOLD file, use the SET HOLDATTR command.

The TITLE attribute has no effect in a report if the column is used with a prefix operator such as AVE. You can supply an alternate column title for columns with prefix operators by using the AS phrase.

If you are using DEFINE fields in a Master File, you must place the TITLE attribute on a line by itself. The semicolon after the DEFINE must appear on the same line as TITLE=.

Syntax **How to Change the Default Column Title**

The syntax is:

```
TITLE='text' , $  
TITLE='text,text,...' , $  
TITLE='text  /' , $
```

where:

text

Is any string that consists of a maximum of 64 characters.

You can split the text across as many as five separate lines. Use single quotation marks to delimit the text, and commas to divide the text into separate lines on the report output.

You can include blanks at the end of an alternate column title by entering a slash (/) in the last position that will be blank, followed by a closing single quotation mark.

The TITLE attribute cannot span more than one line in the Master File. If necessary, move the entire attribute to a line by itself.

Example **Changing the Default Column Title**

The following example shows how to replace the default column title, LNAME, with a title of Client Name.

```
FIELD=LNAME,ALIAS=LN,FORMAT=A15,TITLE='Client Name' , $  
Client Name  
-----
```

Example **Changing the Default Column Title to a Two-line Column Title**

The following example shows how to replace the default column title, LNAME, with a two-line column title of Client Name.

```
FIELD=LNAME,ALIAS=LN,FORMAT=A15,TITLE='Client ,Name' , $  
Client  
Name  
-----
```

Example Controlling the Underline for a Column Title

The following example shows how to control the length of the underline for an alternate column title.

```
FIELD=LNAME,ALIAS=LN,FORMAT=A15,TITLE='Client,Name /', $
Client
Name
-----
```

Example Specifying a Column Title for a DEFINE Field

The following example shows how to replace the default column title for the DEFINE field, ITEMS_SOLD, with a two-line column title of Items Sold.

```
DEFINE ITEMS_SOLD/D8=ORDERED-INVENTORY
;TITLE='Items,Sold', $
Items
Sold
-----
```

Specifying an Online Help Message

The HELPMESSAGE attribute enables you to specify an online help message for a field on a data entry screen. It is available only for FOCUS data maintenance applications.

When a Help Message Displays

Messages specified with the HELPMESSAGE attribute display in the TYPE area of the CRTFORM when you:

- Enter a value for a database field that is invalid according to the criteria defined by the ACCEPT attribute.
- Enter a value for a database field that fails a VALIDATE test.
- Enter a value for a database field that causes a format error.
- Place the cursor in a data entry field and press a help key.

Regardless of the condition that triggers the display of the help message, the same message will appear.

Syntax **How to Specify a Help Message**

The syntax is

```
HELP[MESSAGE]=text , $
```

where:

text

Is one line of text. It can consist of a maximum of 78 characters.

You can use all characters and digits. If the text contains a comma, you must enclose the text in single quotation marks. Leading blanks are ignored.

The HELPMESSAGE attribute cannot span more than one line in the Master File. If necessary, move the entire attribute to a line by itself.

Example **Displaying a Help Message to List Valid Values for a Column**

The following example shows how to display a help message when the DEPARTMENT value is other than MIS, PRODUCTION, or SALES. DEPARTMENT has an ACCEPT attribute which tests values entered for it.

```
FIELDNAME=DEPARTMENT, ALIAS=DPT, FORMAT=A10,  
ACCEPT=MIS PRODUCTION SALES,  
HELMMESSAGE='DEPARTMENT MUST BE MIS, PRODUCTION, OR SALES', $
```

If you enter a value other than MIS, PRODUCTION, or SALES for DEPARTMENT on a CRTFORM, both the default message and help message display on the screen:

```
DATA VALUE IS NOT AMONG ACCEPTABLE VALUES FOR DEPARTMENT  
DEPARTMENT MUST BE MIS, PRODUCTION, OR SALES
```

Example **Displaying a Help Message When a Format Error Occurs**

The following example shows how to display a help message when a format error occurs in HIRE_DATE. Note that the format for HIRE_DATE is integer.

```
FIELDNAME=HIRE_DATE, ALIAS=HDT, FORMAT=I6YMD,  
HELMMESSAGE=THE FORMAT FOR HIRE_DATE IS I6YMD, $
```

If you enter alphanumeric characters for HIRE_DATE on a CRTFORM, both the default message and help message display on the screen:

```
FORMAT ERROR IN VALUE ENTERED FOR FIELD HIRE_DATE  
THE FORMAT FOR HIRE_DATE IS I6YMD
```

CHAPTER 2

Getting Started in Adabas

Topics:

- Preparing the Adabas Environment
- Configuring the Data Adapter for Adabas
- Adabas Overview
- Managing Adabas Metadata
- Overview of Master and Access Files
- Master Files for Adabas
- Access Files for Adabas
- Mapping Adabas Descriptors
- Mapping Adabas Files With Variable-length Records and Repeating Fields
- Using the GROUP Attribute to Cross-Reference Files
- Platform-Specific Functionality
- Customizing the Adabas Environment
- Adabas Reporting Considerations
- Adabas Writing Considerations
- Data Adapter Navigation
- Entry Segment Retrieval of Adabas Records
- Descendant Periodic Groups and Multi-value Fields
- Descendant Adabas Records

The Data Adapter for Adabas allows applications to access Adabas data sources. The adapter converts application requests into native Adabas statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

Preparing the Adabas Environment

Adabas environment variables must be set before you configure the server. Refer to the Software AG documentation for more information about required environment variables.

Procedure How to Prepare the Adabas Environment on Windows NT/2000

On Windows NT/2000, the Adabas environment is set up during the installation of Adabas.

Procedure How to Prepare the Adabas Environment on UNIX

1. Specify the Adabas database directories to access using the UNIX environment variable \$ADADIR. For example:

```
ADADIR=/rdbms/sag/ada
export ADADIR
```

2. Specify the Adabas driver using the UNIX environment variable \$ADALNK. For example:

```
ADALNK=/rdbms/sag/ada/v31135/adalnk.so
export ADALNK
```

3. Specify the version of the Adabas product using the UNIX environment variable \$ADAVERS. For example:

```
ADAVERS=v31135
export ADAVERS
```

Procedure How to Prepare the Adabas Environment on OpenVMS

When a site creates an Adabas ID, the software automatically generates a DCL setup script so users (and products such as iWay) can properly invoke the environment.

The specification for the location of the Adabas setup file is

```
$(SAG$ROOT:[adabas_root]LOGIN.COM
```

where:

```
SAG$ROOT
```

Is the disk on which Adabas is installed.

```
adabas_root
```

Is the root directory of the Adabas installation.

Example Specifying the Data Adapter for Adabas on OpenVMS

```
$(SAG$ROOT:[Adabas]LOGIN.COM
```

The logicals ADABAS\$MAIN, ADALNK and ADABAS\$VERSION will be used to communicate with Adabas Nucleus.

Configuring the Data Adapter for Adabas

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Procedure How to Configure the Data Adapter From the Web Console Using Generic Parameters

The following Adabas parameters are applicable to all platforms and must be entered into the input fields of the Add Adapter screen.

1. Specify the Adabas database number to access. For example:
001
2. Specify the number of the Adabas Employee or another Demo File. For example:
011
3. Click *Configure* after the parameters are entered. Configuration results will appear on the screen. The SET CONNECTION_ATTRIBUTES command is inserted into the edasprof.prf file. For example,

```
ENGINE ADBSINX SET CONNECTION_ATTRIBUTES ;1:ADAEMPL=11
```

Procedure How to Configure the Data Adapter From the Web Console Using OS/390 and z/OS Parameters

The following Adabas parameters are applicable to OS/390 and z/OS platforms only, and must be entered into the input fields of the Add Adapter screen.

1. Specify the Adabas router SVC number. For example:
240
2. Specify the type of the device where the Adabas Associator is located. For example:
3390
3. Specify the name of the Adabas source library that contains member ADALNK. For example:
ADABAS.V713.SRCE
4. Specify the name of the Adabas Associator Data Set. For example:
ADABAS.V713.EXAMPLE.DB001.ASSOR1
5. Click *Configure* after the parameters are entered. Configuration results will appear on the screen. The SET CONNECTION_ATTRIBUTES command is inserted into the edasprof.prf file. For example,

```
ENGINE ADBSINX SET CONNECTION_ATTRIBUTES
;3: "ADAEMPL=1, ADASVC=240, ADADEVICE=3390, ADAASSO=ADABAS.V713.EXAMPLE.DB
001.ASSOR1"
```

Declaring Connection Attributes

No user ID and password is needed to establish a connection to Adabas. This means that Trusted Mode is the standard one for Adabas. To secure a connection, use the SET PASSWORD feature.

Overriding Default Passwords in Specific Files

Passwords for Adabas files can be set in the server profile using the SET PASSWORD command. You can set default passwords for all files and/or databases. Specific passwords can be set for specific files which will override the default. The SET command overrides the password coded in the Access File.

Syntax How to Use the SET PASSWORD Command

The SET PASSWORD command remains valid throughout the user's session.

The syntax is

```
ENGINE ADBSINX SET PASSWORD password FILENO ALL DBNO {ALL|dbno}
ENGINE ADBSINX SET PASSWORD password FILENO fileno DBNO dbno
ENGINE ADBSINX SET PASSWORD OFF
ENGINE ADBSINX SET PASSWORD DEFAULT
```

where:

ADBSINX

Indicates the Adabas data source.

password

Is the password, which can be from one to eight characters in length.

FILENO

Specifies the file number for which the password is set.

DBNO

Specifies the database or databases for which the password is set.

ALL

Indicates all files and/or databases used with the FILENO and DBNO parameters. If you want to use ALL for both FILENO and DBNO, issue this command before any other subsequent password commands. ALL overrides any prior settings.

dbno

Is any valid numeric database value between 0 and 255 (65,535 is the maximum value on a mainframe platform). The DBNO parameter indicates the actual database number.

fileno

Provides a file number, a list of file numbers, and/or a range of file numbers used with the FILENO parameter. Numbers and ranges can be combined by separating items with commas. Valid file numbers range between 0 and 255 (5000 is the maximum value on a mainframe platform).

OFF

Clears all previous ADBSINX SET PASSWORD commands. The Data Adapter for Adabas does not use any passwords specified in the Access File. This command lets you access only those files that have no password security.

DEFAULT

Clears all previous ADBSINX SET PASSWORD commands and causes the Data Adapter for Adabas to use the password in the Access File.

To set a default password:

- Issue the SET PASSWORD command using the ALL keyword for FILENO and/or DBNO.
- Issue specific password requests by specifying particular file and/or database numbers (or ranges) in a subsequent SET PASSWORD command.

Note that subsequent requests are appended to previous requests. Changes are cumulative. The passwords can be issued from a FOCEXEC, which may be encrypted.

Example Using the SET PASSWORD Command to Specify Files in a Database

To set the password to BRUCE for specific files in database number 123 on OS/390 and z/OS, issue the following command:

```
ENGINE ADBSINX SET PASSWORD BRUCE FILENO 1,3-5,23,89-93 DBNO 123
```

To clear all previous ADBSINX SET PASSWORD commands on OS/390 and z/OS, causing the Data Adapter for Adabas to use the password in the Access File, issue the following command:

```
ENGINE ADBSINX SET PASSWORD DEFAULT
```

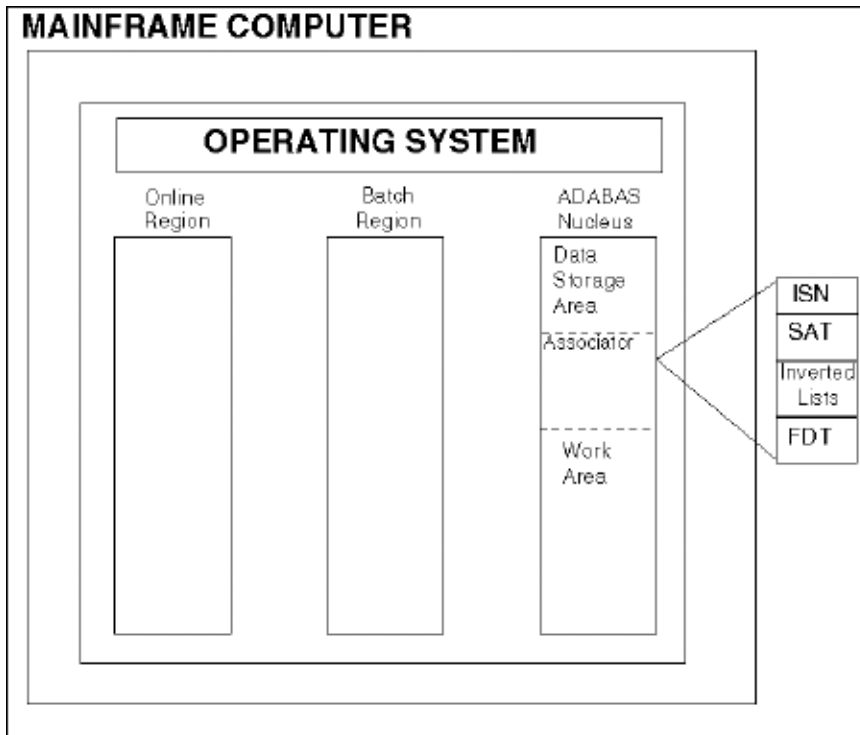
Adabas Overview

This topic provides an overview of Adabas and Adabas files, and includes a discussion of server concepts.

Adabas is a field-oriented DBMS. Data retrievals and updates are performed on a field-by-field basis.

Since Adabas data retrieval occurs at the field level rather than at the record level, your applications may be designed without consideration for the physical organization and maintenance of the record. Data is accessed in a variety of ways (in physical sequence, in logical sequence, or by Internal Sequence Numbers), thereby enabling you to tailor data access to address your specific needs.

The Adabas DBMS consists of several components. The following is an illustration of the Adabas environment.



The following are the components of the Adabas environment:

Operating System	Is a set of programs (software) that control the operation of the computer (hardware).
Online Region	Is part of the Dynamic Area of the computer. It is the section in which multiple jobs execute simultaneously. The Online Region is also known as the Foreground Region (applies only to TSO).
Batch Region	Is also part of the Dynamic Area of the computer. Jobs become a series of commands that are grouped together and processed in batches (applies only to batch jobs).
Data Storage Area	Contains the actual data. The data is stored in compressed form.
Associator	Stores relationships about the data. It contains the following components: <ul style="list-style-type: none"> • Internal Sequence Number (ISN) list, also called the Address Converter, used to determine the physical location of data. • Space Allocation Tables (SATs) to control storage space. • Inverted lists for the defined descriptors. Inverted lists and descriptors are discussed in <i>Inverted Lists: The Adabas Key Structure</i> on page 2-14. • Field Definition Tables (FDTs), which contain detailed data structure information for each field and descriptor.
Work Area	Is a "scratch pad" used by Adabas to build ISN lists and to sort and store records that your program requires.

The Multi-Programming Module (MPM), the cornerstone of the Adabas Nucleus, contains the logic that enables multiple programs, both batch and online, to access Adabas databases simultaneously.

Each database has its own data storage area, associator, and work area, in addition to its own unique database number.

Adabas Files

When accessing Adabas, the server uses logical files as the unit of data retrieval. An Adabas file is identified by a file number. The number is unique within each database.

All file and field documentation for the Adabas database can be stored in the Predict dictionary. The information about the data structure includes processing uses, file ownership, and field descriptions.

Inverted Lists: The Adabas Key Structure

An Adabas file is defined with a set of indexes called descriptors. The Adabas term for these values, which reside in the File Associator Table, is inverted lists. Inverted lists contain the values of descriptors, a count of the total number of records in which each value appears, and the Internal Sequence Numbers (ISNs), in ascending order, associated with each occurrence.

Adabas utilities create and maintain an inverted list for each descriptor identified. These lists store data in an ascending sequence by the value of the key. A single file may have up to 256 inverted lists associated with it (the number of inverted lists is unlimited for a mainframe platform).

The descriptors are identified to speed data location and to retrieve specific, frequently required data from Adabas files. Of the available types of inverted lists, the Data Adapter for Adabas supports the following:

Inverted List	Description
Descriptors	Key values associated with a single field (also called elementary field descriptors).
Subdescriptors	Key values associated with part of a single field.
Superdescriptors	Key values associated with all or part of two to twenty fields.
Phonetic descriptors	Alphabetic values associated with the first twenty bytes of the field value.
Hyper descriptors	A value generated, based on a user-supplied algorithm.

In the following text, "descriptor" refers to all five types of descriptors interchangeably, unless otherwise indicated.

Two or more files may be related in these ways:

- The database administrator may couple several files based on a common descriptor key in each file.
- An application program may logically relate files if there are fields in one file that can be used to access a key in another.

Field Definition Tables

Field description information for Adabas files is maintained in the FDT, which is part of the Adabas associator. Adabas uses this information when accessing files. The FDT includes the following information:

Field Information	Description
Field Level Indicator	Denotes a hierarchical relationship between fields. In an Adabas record description the levels are 1, 2, 3, and so on.
Adabas Field Name	A two-character name used by Adabas to identify the field. This name is unique to the file. The first character must be alphabetic, and the second character can be alphabetic or numeric. Field names E0 through E9 are reserved for internal Adabas use.
Standard Length	Indicates the length of the field in bytes.
Standard Format	Indicates the format of the field. Refer to the table in <i>Standard Formats for Adabas Fields</i> on page 2-16 for the acceptable formats and their meanings.
Field Properties	Indicates whether the field value is null suppressed (NU), or the field is of fixed or variable length (FI), if the field (which exists on a mainframe platform) is long alphanumeric (LA).
Descriptor Status	Indicates if the field is a descriptor, subdescriptor, superdescriptor, phonetic descriptor, hyper descriptor, subfield, or superfield.
Field Type	Indicates if a field is a group (GR), multi-value (MU), or periodic group (PE). Multi-value fields and periodic groups are examples of multiply occurring fields.

Typically, an Adabas FDT contains some, but not all, of the field characteristics discussed above. To view the FDT, Software AG provides the ADAREP report. See your Software AG documentation for more information on how to run the ADAREP report.

The external field name is in the Predict dictionary.

External Field Name	A 32-byte name used in a Predict to identify the field. This name is unique to the file.
----------------------------	--

Reference Standard Formats for Adabas Fields

The following table shows the acceptable standard formats for Adabas fields and their meanings:

Format	Description
A	Alphanumeric data field with a maximum of 253 bytes (126 for descriptor fields).
U	Zoned decimal data field with a maximum of 29 bytes (signed, unpacked data).
P	Signed packed decimal data field with a maximum of 15 bytes.
B	Unsigned binary data field with a maximum of 126 bytes.
F	Single-precision, floating point data field that is always four bytes long.
G	Decimal, double-precision integer field with an eight-byte maximum.

In the following example, the column on the left illustrates a partial Adabas FDT. The bold numbers on the left refer to the numbered annotations that follow:

PAY-FILE

FIELD DESCRIPTION (from FDT)

FIELD NAME (from DDM)

1. 01, PS, 08, P, NU, DE
 2. 01, PW, 04, P
 3. 01, MI, PE
 02, MH, 04, P
 02, MW, 04, P
 02, MT, 04, P
 4. 01, PT, 04, P, DE, MU
 01, PC, 08, P, DE, MU

SSN
 HOURLY_WAGE
 MONTHLY_INFO
 MONTHLY_HOURS
 MONTHLY_WAGES
 MONTHLY_TAX
 TAX
 CHILDS_SSN

1. The first field, SSN, is an elementary-level field. Its internal name is PS, it is eight bytes long, and it is in packed decimal data format. The field is null suppressed, as indicated by the NU, and it is a descriptor (DE).
2. HOURLY_WAGE is also an elementary-level field. It is called PW internally, is four bytes long, and is also in packed decimal data format.

3. MONTHLY_HOURS, MONTHLY_WAGES, and MONTHLY_TAX are all subordinate fields of the MONTHLY_INFO periodic group (PE). Internally, they are called MH, MW, and MT, respectively. Each is four bytes long and is in packed decimal data format.
4. TAX and CHILD_SSN are elementary-level fields. TAX is called PT internally, is four bytes long, is in packed decimal data format, and is a multi-value (MU) field with a descriptor (DE) associated with it. CHILD_SSN is called PC internally, is eight bytes long, is in packed decimal format, and is also a multi-value field with a descriptor associated with it.

For more information about Adabas periodic groups (PE) and multi-value (MU) fields, see *Mapping Adabas Files With Variable-length Records and Repeating Fields* on page 2-62.

Managing Data Storage With Null-Suppression

Null-suppression is one of the methods Adabas employs to manage data storage. Only the fields within a record that contain data are stored. Fields containing blanks for alphabetic characters or zeros for numeric data are not stored. They are represented by a one-byte "empty field" character. When a procedure accesses a record containing a null-suppressed field, Adabas expands the field to its full length and includes the null values of blanks or zeros.

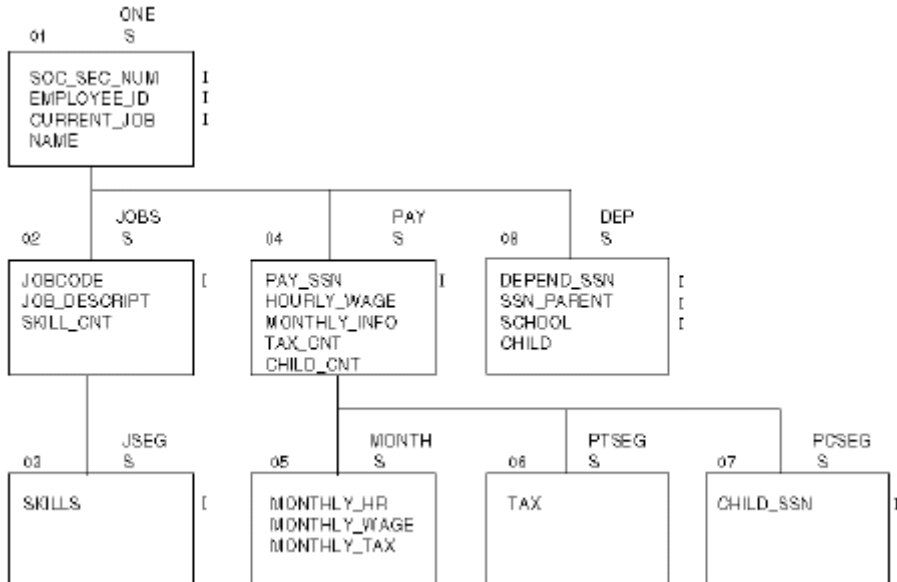
When the null-suppressed field is a descriptor or part of a superdescriptor, no entry is made for the record containing that field in the inverted list. It would not be productive to have the inverted list direct you to records in which the descriptor has no data.

A field with null-suppression appears with the NU attribute in the Adabas FDT.

Server File Structure

The server uses a non-procedural language to create reports, graphs, and extract files. It also enables you to access data files without knowing the details of the file structure or access method.

The server treats any data file as either a single-path or multi-path hierarchy. Graphically, information is laid out using an inverted tree structure, as in the following sample STAFF file. In the example, the letter I to the right of a field indicates an indexed field.



The most general information appears at the top, and the more specific information appears under it. Each box in the structure is referred to as a segment. A database can consist of one or more logically related segments.

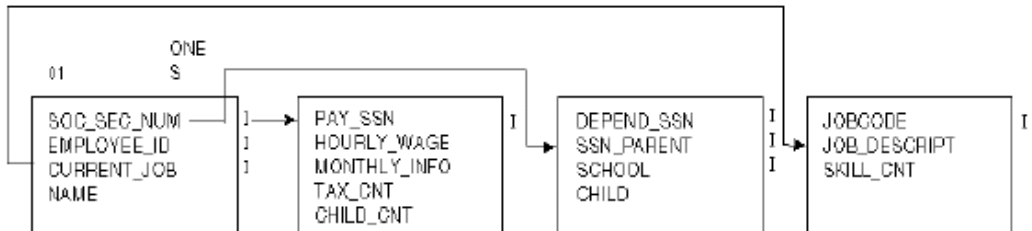
When logically related information is retrieved using this database structure, multiple occurrences of each segment are created. Each occurrence of a segment is called a segment instance. Each Adabas database is equivalent to a collection of logically related segment instances.

The retrieval sequence of the segments is determined by the view of the structure, that is, the order of the segments from top to bottom and left to right. The segment at the top is the parent, or root, segment. The segments under the parent are the child, or descendant, segments.

Generally, parent and child segments have a one-to-many relationship; that is, a single parent has multiple occurrences of one child segment. However, the server also handles one parent-one instance only of a child segment.

Adabas uses specific concepts and techniques for organizing data. It holds data in logically distinct files that are interrelated using fields that share common formats and values called descriptors.

Within a single Adabas file, records that contain multiple occurrences of a field can vary in length and format. The following example illustrates an Adabas structure containing four separate files that are linked by common fields:



Many Adabas structures are more complex than the preceding example.

Adabas Descriptors

Descriptors are fields, partial fields, or groups of complete and partial fields used by Adabas to select records in a file. Adabas descriptors correspond to indexed fields.

There are five types of descriptors supported by the Data Adapter for Adabas:

- Descriptors (DE), which have a value associated with a single field.
- Superdescriptors (SPR), which have a key value associated with all or part of two to twenty fields (see your Software AG documentation for current limitations on superdescriptors).
- Subdescriptors (NOP), which have a key value associated with part of a single field.
- Phonetic Descriptors (PDS), which have a similar phonetic value associated with a single field. The phonetic value of a descriptor is based on the first twenty bytes of the field value. Only alphabetic values are considered: numeric values, special characters, and blanks are ignored. Lower- and uppercase alphanumeric characters are internally identical.
- Hyper Descriptors (HDS), which have a value generated, based on a user-supplied algorithm.

All five types of descriptors are collectively referred to as "descriptors."

Consider the Adabas FDT for the STAFF file:

STAFF FILE FDT

FIELD DESCRIPTION (from FDT)	FIELD NAME (from Predict)
01, ES, 08, P, NU, DE	SOC_SEC_NUM
01, EI, 06, A, NU, DE	EMPLOYEE_ID
01, EJ, 03, A, DE	CURRENT_JOB
01, EN, 24, A	NAME

Notice that SOC_SEC_NUM, EMPLOYEE_ID, and CURRENT_JOB are labeled as descriptors. Any one of these three fields can be used to search the STAFF file.

Adabas descriptors, superdescriptors, and subdescriptors must be declared in Master and Access Files in certain ways. For more information, see *Mapping Adabas Descriptors* on page 2-54.

Adabas Files With Fixed-length Records

Unless an Adabas file has multi-value fields or periodic groups, each field occurs once in each record. When describing an Adabas file, you do not need to describe all the fields in your Master File; just the fields you use. Note that if you choose a periodic group, all fields in the periodic group must be defined. A single simple Adabas file maps to a single segment on the server, and each field you use in the Adabas file becomes a field in the segment.

Adabas RECORD		SEGMENT	
DB	Name (in Predict)	Field Name	Alias
ES	SOC_SEC_NUM	SOC_SEC_NUM	ES
EI	EMPLOYEE_ID	EMPLOYEE_ID	EI
EJ	CURRENT_JOB	CURRENT_JOB	EJ
EN	NAME	NAME	EN

Adabas Files With Variable-length Records and Repeating Fields

There are two types of Adabas repeating fields:

- Multi-value (MU) fields.
- Periodic (PE) groups.

An MU field is a single field that occurs a variable number of times in a record. It appears as type MU in an Adabas record description. Adabas supports up to 191 occurrences of MU fields per record.

A PE group is a group of contiguous fields that occur a variable number of times in a single record. It appears as type PE in an Adabas record description. Adabas supports up to 191 occurrences of PE fields per record. These component fields are regular fields or MU fields.

Note: The Data Adapter for Adabas supports the maximum number of occurrences of MU fields and PE groups allowed by Software AG.

When an Adabas file contains MU fields or PE groups, the repeating information occurs a variable number of times. The physical record length varies depending on how many times the repeating information actually appears.

More than one segment is needed to accurately describe an Adabas file with repeating data. The number of times the fields repeat is expressed in a counter field. For more information, see *Mapping Adabas Files With Variable-length Records and Repeating Fields* on page 2-62.

Managing Adabas Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Adabas data types.

For the Data Adapter for Adabas to access Adabas files, you must describe each file you use in a Master and Access File. The logical description of an Adabas file is stored in a Master File, which describes the field layout.

Creating Synonyms

Synonyms define unique names (or aliases) for each Adabas table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File, which represent the server's metadata.

The CREATE SYNONYM command automatically generates Master and Access files for Adabas files based on information stored in the Field Definition Table (FDT) in Adabas and the Predict dictionary (optionally). The command requires Adabas release 5.0 or higher and Predict release 3.1.4 or higher.

The syntax is identical on UNIX, Windows, OS/390 and z/OS, and OpenVMS platforms.

Note: You can add the following parameters to the CREATE SYNONYM syntax:

- ISN provides support for retrieval of Adabas internal sequence numbers.
- GFBID allows the use of Adabas Global Format buffers for requests that utilize the same field lists.

Syntax

How to Create a Synonym Manually

```
CREATE SYNONYM appname/synonym
FOR [FILE=file-number[/predict-file-number(predict-filename)]]
DBMS ADBSINX
DATABASE dbid[/predict-dbid]
PARMS '<parameter_list>'
END
```

where:

appname

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, then this application name must not be used.

synonym

Is the alias for the data source (maximum 64 characters).

file-number

Is the number of the Adabas file used as the base for the Master and Access Files. The range is from 1 to 255 (5000 is the maximum value on a mainframe platform).

predict-file-number

Is the number of the Adabas file where the Predict dictionary data is located. The range is from 1 to 255 (5000 is the maximum value on a mainframe platform).

predict-filename

Is the name of the view in the Predict dictionary that corresponds to the Adabas file being used. Use this name when it differs from *synonym*. Maximum length is 32 bytes.

ADBSINX

Is the SUFFIX that indicates the Adabas data source.

dbid

Is the Database ID (DBNO) for the Adabas DBMS to be accessed. The range is from 1 to 255 (65535 is the maximum value on a mainframe platform).

predict-dbid

Is the Database ID (DBNO) for the Adabas DBMS that contains the Predict dictionary data. The range is from 1 to 255 (65535 is the maximum value on a mainframe platform).

parameter_list

Is the parameter values delimited by space or comma. Possible values are:

GF specifies that a field generated in the Master File is to be used to specify Adabas Global Format Buffer ID values.

IS specifies that a field generated in the Master File is to be used to specify Adabas Internal Sequence Numbers (ISN).

NEW or **STD** is the processing mode for Adabas superdescriptor fields. For more information, see *NEW Synonym Setting for Superdescriptors* on page 2-72.

Note: These values override values that were set by SET command. For more information, see *Customizing the Adabas Environment* on page 2-72.

Example **Creating Synonyms for Adabas With and Without Predict Existences**

- **When the Predict file exists:** Issue CREATE SYNONYM with the name SAMPLE for file 2 from Database 1. Predict dictionary is located in file 12 from database 2. The name of the view from the Predict dictionary is VEHICLES:

```
CREATE SYNONYM SAMPLE FOR FILE=2/12(VEHICLES)
DBMS ADBSINX DATABASE 1/2
END
```

- **With the Predict file and the same file name:** Issue CREATE SYNONYM with the name SAMPLE for file 2 from Database 1. The name of the view from the Predict dictionary is same as the Master and Access files:

```
CREATE SYNONYM SAMPLE FOR FILE=2/11
DBMS ADBSINX DATABASE 1/1
END
```

- **Without the Predict file:** Issue CREATE SYNONYM as follows:

```
CREATE SYNONYM FILE12 FOR FILE=12
DBMS ADBSINX DATABASE 1
END
```

Below are samples of Master and Access Files generated in this example.

Generated Master File: file12.mas

```
FILE=FILE12 ,SUFFIX=ADBSINX , $

SEGNAME=S01 ,SEGTYPE=S0 , $
FIELD =AA_FIELD ,ALIAS=AA ,A15 ,A15 ,INDEX=I , $
FIELD =AB_FIELD ,ALIAS=AB ,I9 ,I4 , , $
FIELD =AC_FIELD ,ALIAS=AC ,A8 ,A8 ,INDEX=I , $
GROUP =CD_GROUP ,ALIAS=CD ,A50 ,A50 , , $
FIELD=AD_FIELD ,ALIAS=AD ,A20 ,A20 ,INDEX=I , $
FIELD=AE_FIELD ,ALIAS=AE ,A20 ,A20 , , $
FIELD=AF_FIELD ,ALIAS=AF ,A10 ,A10 ,INDEX=I , $
FIELD =AG_FIELD ,ALIAS=AG ,P2 ,Z2 , , $
FIELD =AH_FIELD ,ALIAS=AH ,A1 ,A1 ,INDEX=I , $
FIELD =AI_FIELD ,ALIAS=AI ,A1 ,A1 , , $
FIELD =AJ_FIELD ,ALIAS=AJ ,P6 ,Z6 , , $
$GRMU =AK_GROUP ,ALIAS=AK ,A7 ,A4 , , $
FIELD=AL_FIELD ,ALIAS=AL ,A3 ,A3 , , $
FIELD=AM0101_CNT ,ALIAS=AMC ,I4 ,I1 , , $
$ $$$$$$$$$$$$$$$$$$$$$ Superdescriptor $$$$$$$$$$$$$$$$$$$$$ $
FIELD =AN_FIELD ,ALIAS=AN ,A4 ,A4 ,INDEX=I , $
$ FIELD=AJ_FIELD_S01 ,ALIAS=AJ ,P6 ,Z2 , , $
$ FIELD=AJ_FIELD_S01 ,ALIAS=AJ ,P6 ,Z2 , , $
$ $$$$$$$$$$$$$$$$$$$$$ Superdescriptor $$$$$$$$$$$$$$$$$$$$$ $
GROUP =AO_GROUP ,ALIAS=AO ,A28 ,A22 ,INDEX=I , $
FIELD=AG_FIELD_S02 ,ALIAS=AG ,P2 ,Z2 , , $
FIELD=AD_FIELD_S02 ,ALIAS=AD ,A20 ,A20 ,INDEX=I , $

SEGNAME=AM0101 ,SEGTYPE=S0 , PARENT=S01 ,OCCURS=AMC , $
FIELD =AM_FIELD ,ALIAS=AM ,P7 ,P4 , , $
FIELD =AM0101_OCC ,ALIAS=ORDER ,I4 ,I1 , , $
```


Generated Access File: file12.acx

```

RELEASE=6      ,OPEN=YES                                     , $
SEGNAM=S01     ,ACCESS=ADBS ,FILENO=12    ,DBNO=1      ,WRITE=YES     ,
                UNQKEYNAME=AA_FIELD      ,
                CALLTYPE=FIND             , $
$              CALLTYPE=RL  ,SEQFIELD=AA_FIELD           , $
$FIELD= AA_FIELD                                     ,TYPE=DSC     , $
$FIELD= AC_FIELD                                     ,TYPE=DSC     , $
$FIELD= AD_FIELD                                     ,TYPE=DSC     , $
$FIELD= AF_FIELD                                     ,TYPE=DSC     , $
$FIELD= AH_FIELD                                     ,TYPE=DSC     , $
FIELD= AN_FIELD                                     ,TYPE=NOP     , $
FIELD= AO_GROUP                                     ,TYPE=SPR     , $
FIELD=AG_FIELD_S02                                 ,TYPE=       ,NU=YES     , $
FIELD=AD_FIELD_S02                                 ,TYPE=DSC    ,NU=YES     , $
SEGNAM=AM0101 ,ACCESS=MU    ,FILENO=12    ,DBNO=1      ,WRITE=YES     , $

```

Example Creating a Synonym Using a Parameter List

Issue the following command to create Master and Access Files with ISN support and to process a superdescriptor in the NEW mode:

```

CREATE SYNONYM SAMPLE FOR 12
DBMS ADBSINX DATABASE 1
PARMS 'ISN, NEW'
END

```

Note:

- Only one Adabas file description can be generated in the Master File. If multiple Adabas files must be described in one Master File, a description for each Adabas file must be created and combined in one Master File manually.
- The CREATE SYNONYM command describes Adabas files within the constraints of the Data Adapter for Adabas. For more information, see *Platform-Specific Functionality* on page 2-71. Superdescriptors, subdescriptors, periodic elements (PE groups), and multi-value fields (MU) are included. Comments are used to describe unsupported fields in the generated Master File.
- The CREATE SYNONYM command allows you to customize the output using data from the Predict dictionary. You can:
 - Replace generic field names constructed based on Adabas field names (xx_FIELD) with the names obtained from the Predict dictionary.
 - Replace the USAGE format and length generated from an Adabas FDT (Field Definition Table) with the format and length from the Predict dictionary.
 - Include the NATURAL column heading stored in the Predict dictionary in the Master File for use as the default column heading in server reports.

Reference Master File Field Attributes

The following field attributes describe Adabas data segments.

Field Attribute	Description
FILE	Is the Master File name. It may or may not match the file name in the Adabas DBMS.
SUFFIX	Is always ADBSINX.
SEGNAME	Are the segment names in the description generated by CREATE SYNONYM. They follow a logical format to provide uniqueness within the file.
FIELD=	Is the field name from the Predict dictionary (if used) or generated automatically (xx_FIELD).
GROUP=	Identifies fields described as simple groups or PE groups. Is the field name from the Predict dictionary (if used) or generated automatically (xx_GROUP).
ALIAS=	Is the actual Adabas short name. It can be detailed for counter fieldcounter fields without standard length. For order fields, it has the value ORDER.
USAGE=	Is the USAGE format and length of the field. This attribute determines how the value is displayed in reports. Values are determined based on ACTUAL format and length and then may be detailed by value from the Predict dictionary (if used). For fields without standard length, it has the maximal value for the format.
ACTUAL=	Is the field standard format and length as stored in the Adabas FDT (Field Definition Table) for a given file. For fields without standard length, it has the maximal value for the format. For fields with format A and option LA (Long Alphanumeric, used only for OS/390 and z/OS) the ACTUAL length must have a value greater than 253. Value 500 is assumed.
INDEX=I	Indicates the field is a superdescriptor, subdescriptor, or simple descriptor.
TITLE=	Is the column heading used in reports. This attribute is included only if the Predict dictionary is used and the field has a column heading value in the Predict dictionary.

Field Attribute	Description
GROUP=	Identifies fields described as simple groups or PE groups.
\$GRMU=	Is a group that contains a PE group or an MU (multi-value) field.
\$2LONG=	Is a group field whose total length exceeds the maximum of 4096 characters supported by the server.
\$PEMU=	Is a PE group that contains an MU (multi-value) field or is a group field that contains a PE group or an MU field. In both cases, the field cannot be used to retrieve data using the adapter.
\$FIELD=	Is the field that belongs to a commented group or super descriptor composed of partial fields (NOP).

Reference Access File Attributes

The following Access File attributes describe Adabas data segments:

Access File Attribute	Description
RELEASE=	Indicates the Adabas release. If RELEASE=5 or less, then FILENO can be 1–255 and DBNO can be 0–255. If RELEASE is 6 or greater, then the adapter supports two-byte FILENO (1–5000) and DBNO (1–65535).
OPEN=	Determines whether the adapter should issue Adabas OPEN and CLOSE calls for each report request.
SEGNAM=	Is the segment name as described in the Master File.
ACCESS=	Specifies the access method for the segment. ADBS indicates segments that contain non-repeating data. PE indicates segments that describe periodic groups. MU indicates segments that describe multi-value fields.
DBNO=	Specifies the Adabas database number. On UNIX, Windows, and OpenVMS, this attribute must be included in the Access File. DBNO depends on RELEASE. If RELEASE is 5 or less, DBNO can be 0–255; if RELEASE is 6 or greater, DBNO can be 1–65535.
FILENO=	Is the Adabas file number. If RELEASE is 5 or less, FILENO can be 1–255; if RELEASE is 6 or greater, FILENO can be 1–5000.

Access File Attribute	Description
UNQKEYNAME=	Is the field name that will be used as a unique key during file updating. The first unique indexed field from the root segment is used. If that does not exist, then a value from SEQFIELD is used.
CALLTYPE=	FIND. Creates an additional commented line with CALLTYPE=RL if the Predict file view contains SEQFIELD data or the root segment contains an indexed field. The line with the preferable value of CALLTYPE can be chosen depending on the method used to retrieve the data from the database.
SEQFIELD=	Is the SEQFIELD name taken from the Predict file view, if used, or the first unique indexed field, or the first indexed field from Root segment. If CALLTYPE=FIND, then no value is entered for SEQFIELD in the Access File.
FIELD=	Describes a descriptor, superdescriptor, subdescriptor, phonetic descriptor, hyper descriptor, or a component of a superdescriptor.
TYPE=	Indicates the type of descriptor for FIELD. Values are SPR for superdescriptors, NOP for subdescriptors or superdescriptors composed of partial fields, PDS for phonetic descriptors, HDS for hyper descriptors, or DSC for a simple descriptor.
NU=	Indicates if the field uses null suppression. Values are YES or NO.
PASS=	Is the Adabas password for the file. This attribute can be added manually.
KEYFLD=	Is for child segments only. This is the common field in the parent segment, which is used to relate the two files.
IXFLD=	Is for child segments only. This is the common field in the child segment, which is used to relate the two files.

Reference **Comments in Master and Access Files**

The generated Master File and Access File contain several commented entries. They are provided for the client's convenience and are not used by the server. Comments are provided for:

- The CREATE SYNONYM time stamp and creator's User ID (included in both the Master and Access Files).
- The Predict file name, file number, and DBID, if Predict is used.
- The headers before special fields: super/sub/hyper/phonetic descriptors or fields without standard length.
- The fields not supported by the adapter.
- The component fields of super descriptors that are composed of partial fields (described as NOP type).

Syntax **How to Use the ISN and GFBID Parameters With CREATE SYNONYM**

```
CREATE SYNONYM appname/synonym
FOR FILE=file-number[/predict-file-number[/predict-filename]]
DBMS ADBSINX
DATABASE dbid[/predict-dbid][PARMS|GFBID|ISN]
END
```

where:

GFBID

Specifies that a field generated in the Master File is to be used to specify Adabas Global Format Buffer ID values.

ISN

Specifies that a field generated in the Master File is to be used to specify Adabas Internal Sequence Numbers (ISN).

For complete CREATE SYNONYM syntax, see *How to Create a Synonym Manually* on page 2-22.

Reference Usage of Master File Fields With ISN Support

The Data Adapter for Adabas can employ a new data retrieval strategy through Read Logical by ISN (L1) calls. It can be used to determine the Internal Sequence Number (ISN) of a record that was read or that will be inserted into an Adabas file.

ISN-based access is applicable only if an ISN field is described in the Master File. This field has a field name that is user-defined and an ALIAS of ISN. The Usage format is I and the Actual format must be I4. This field can be defined only in segments that contain non-repeating data (that is, using an access method defined in the Access File as ADDBS):

```
FIELD=ISN_FIELD, ALIAS=ISN, I10, I4, $
```

Equality tests on the ISN field can be used to retrieve a single record when:

- The report request contains an equality operator in the selection test on an ISN list.
or
- The ISN field is used as the cross-referenced field in the Join to an Adabas file.

Adabas returns the Response Code 113 if the record with the ISN defined in the test is not present in the Address Converter for the file. The adapter returns the following message: "Record is not found".

The Data Adapter for Adabas uses Read Logical by ISN (L1) calls to retrieve all records for the entry segment when:

- The report request does not contain optimizable selection criteria on an inverted list.
- The Access File contains a SEQFIELD value for the segment whose value is equal to the ISN field name.

For example, the Access File ADATEST contains:

```
SEGNAM=S01, ACCESS=ADBS , ... ,SEQFIELD=ISN_FIELD , $
```

Adabas returns each record in ascending order by the value of the ISN.

Example Equality Test on ISN Field

```
SELECT AA_FIELD, AJ_FIELD FROM ADATEST  
WHERE ISN_FIELD = 1100;
```

Note: Multifetch option will be suppressed for this call.

Read Logical by ISN (L1) calls to retrieve a subset of records for the entry segment when:

- The report request contains the GE/GT relational operator in the selection test on an ISN list.
- The report request doesn't contain any selection tests on an inverted list.
- The Access File contains a SEQFIELD value for that segment.

Example Inequality Test on ISN Field

AFD ADATEST contains:

```
SEGNAM=S01, ACCESS=ADBS , ... ,SEQFIELD=AA_FIELD , $
```

Read Logical by ISN (L1) is used:

```
SELECT AA_FIELD, AE_FIELD FROM ADATEST
WHERE ISN_FIELD > 1100;
```

Reference ISN for Insert

After issuing an Insert request on a file with an ISN field in the Master File, the resulting ISN generated by Adabas is displayed by the message FOC4592:

```
(FOC4592) RECORD IS INSERTED WITH ISN : nnnnn
```

Example Issuing an Insert Request

SQL

```
INSERT INTO ADBTEST (AA_FIELD, AJ_FIELD)
VALUES ('11111111', 'TAMPA');
END
```

```
ADBTEST ADBSINX ON 09/20/2002 AT 15.22.16
(FOC4592) RECORD IS INSERTED WITH ISN : 11
(FOC4566) RECORDS AFFECTED DURING CURRENT REQUEST : 1/INSERT
```

```
TRANSACTIONS: TOTAL = 1 ACCEPTED= 1 REJECTED= 0
SEGMENTS: INPUT = 1 UPDATED = 0 DELETED = 0
```

You can assign a value for the ISN field if the Insert request contains the ISN field and the assigned value is not 0.

The adapter will issue an Adabas N2 Direct Call to assign this ISN value to the inserted record. Adabas returns Response Code 113 if this value was already assigned to another record in the file or if it is larger than the MAXISN in effect for the file.

Reference Master File Fields and GFBID Support

The Data Adapter for Adabas can optimize the performance of queries that use the same Adabas field lists repeatedly. Field lists are generated in Adabas format buffers, and can be retained.

Global Format Buffer ID (GFBID) support is applicable only if a GFBID field is present in the Master File. This field has a user-defined field name, and an ALIAS of GFBID. This field is used to determine the Global Format Buffer ID that will be defined in read requests with identical field lists for the same database. The GFBID field has a Usage format of A8 and an Actual format of A8. This field can be defined only in segments that contain non-repeating data (that is, the access method defined in the Access File is ADBS).

Example Using GFBID Syntax in a Master File

```
FIELD=GID_FIELD, ALIAS=GFBID, A8, A8, $
```

Note:

- If the field list is changed but the same GFBID is used in a request, then incorrect results may be displayed. In some cases, an error message about the possible mismatch between the Field Definition Table (FDT) and Master File will be issued.
- If two requests have the same list of selected fields but different fields are used in selection criteria, then the requests must use different GFBID values.

The GFBID field can be defined in the Master File manually or using the Create Synonym facility with option PARMS GFBID.

Example Using the GFBID Parameter in the CREATE SYNONYM Command

```
CREATE SYNONYM ADATEST FOR 11 DBMS ADBSINX DATABASE 3 PARMS 'GFBID'  
END
```

The value of GFBID can have up to 8 bytes and must start with a digit or uppercase character.

If GFBID is used in a request against a file that contains simple PE/MU segments, it will be applied to Adabas calls to get the field values from these segments. The GFBID value will be adjusted for each non-ADBS segment; that is, a segment number is placed in the last (8th) byte of the GFBID field.

Calls to a PE (periodic element) with MU (multi-value) child or an MU with PE parent segment will not use GFBID values.

Note: To make the GFBID value unique within Adabas, do not use 8 byte GFBID values for these types of calls.

- If a GFBID field is defined in the Master File, it can be used in a request as part of the selection test. The adapter will take the GFBID value from the selection criteria, deactivate the field, and remove it from the match array. The GFBID value is placed in the ADD5 field of the Adabas Control Block for the request.
- If the GFBID is applied to an Adabas call, then field name ADD5 appears in the adapter trace. GFBID values that were issued in all requests to the database are accumulated in the Adabas internal queue. These values can be removed from the queue by issuing the following adapter SET command:

```
ENGINE ADBSINX SET GFBID_OFF ALL/<value> DBID <number>
```

When a single <value> is in the SET command, all values issued for child segments are removed as well. If <value> contains blanks, then value must be in single quotes.

ALL clears all GFBID values.

Example Using GFBID Syntax

MFD ADATEST contains:

```
FIELD=GID_FIELD, ALIAS=GFBID, A8, A8, $
```

Request 1. Global ID = ABC1:

```
SQL
SELECT AA_FIELD, AJ_FIELD, ISN_FIELD, AQ0201_OCC, AR_FIELD, AS_FIELD,
AT_FIELD
FROM EMPLOYEES
WHERE ISN_FIELD > 1100 AND GID_FIELD = 'ABC1';
END
```

Result:

PAGE 1

AA_FIELD	AJ_FIELD	ISN_FIELD	AQ0201_OCC	AR_FIELD	AS_FIELD	AT_FIELD
30034517	DERBY	1105	1	UKL	6500	750
30034517	DERBY	1105	2	UKL	6800	1240
30034517	DERBY	1105	3	UKL	7100	1345
30034517	DERBY	1105	4	UKL	7450	1450
30034517	DERBY	1105	5	UKL	7800	1700
50005600	VIENNE	1106	1	FRA	165810	10000
50005300	PARIS	1107	1	FRA	166900	5400

NUMBER OF RECORDS IN TABLE= 7 LINES= 7

Request 2. Global ID = ABC1, but field's list is changed:

```
SQL
SELECT AA_FIELD, ISN_FIELD, AQ0201_OCC, AR_FIELD, AS_FIELD, AT_FIELD
FROM EMPLOYEES
WHERE ISN_FIELD > 1100 AND GID_FIELD = 'ABC1';
END
```

Result:

```
(FOC4567) POSSIBLE MISMATCH BETWEEN FDT AND MFD : file=19879548
(FOC4591) ERROR RETURN ON READ BY ISN : S01 /999:ffffffcd001d00
```

```
NUMBER OF RECORDS IN TABLE= 0 LINES= 0
```

Request 3. Remove value 'ABC1' from Adabas:

```
ENGINE ADBSINX SET GFBID_OFF ABC1 DBID 3
```

Request 4. Repeat request 2.

```
SQL
SELECT AA_FIELD, ISN_FIELD, AQ0201_OCC, AR_FIELD, AS_FIELD, AT_FIELD
FROM EMPLOYEES
WHERE ISN_FIELD > 1100 AND GID_FIELD = 'ABC1';
END
```

Result:

```
PAGE 1
```

AA_FIELD	ISN_FIELD	AQ0201_OCC	AR_FIELD	AS_FIELD	AT_FIELD
30034517	1105	1	UKL	6500	750
30034517	1105	2	UKL	6800	1240
30034517	1105	3	UKL	7100	1345
30034517	1105	4	UKL	7450	1450
30034517	1105	5	UKL	7800	1700
50005600	1106	1	FRA	165810	10000
50005300	1107	1	FRA	166900	5400

```
NUMBER OF RECORDS IN TABLE= 7 LINES= 7
```

Request 5. Remove all GFBID's values for Database 3 from Adabas:

```
ENGINE ADBSINX SET GFBID_OFF ALL DBID 3
```

Overview of Master and Access Files

The server processes a report request with the following steps:

1. It locates the Master File for the Adabas file. The filename in the report request is the same as the OS/390 and z/OS PDS member name.
2. It detects the SUFFIX attribute, ADBSINX, in the Master File. Since this value indicates that the data resides in an Adabas file, it passes control to the data adapter.
3. The adapter locates the corresponding Access File and uses the information contained in both the Master and Access Files to generate the Adabas direct calls required by the report request. It passes these direct calls to the Adabas DBMS.
4. The adapter retrieves the data generated by the Adabas DBMS and returns control to the server. For some operations, the server may perform additional processing on the returned data.

Master Files for Adabas

Standard Master File attributes are generally used to describe Adabas files, but there are concepts in Adabas file processing that require special terminology at the field and segment level. On OS/390 and z/OS, the Master File is a member of a PDS with DDNAME MASTER.

The server permits a great deal of flexibility in its Master Files. For the Data Adapter for Adabas, you need to describe only the Adabas fields actually used in reporting applications, omitting any unnecessary Adabas fields. (Note the exception that if a PE group is included in the Master File, all fields of the PE group must be included. See *Field Attributes in Master Files* on page 2-40 and *Describing Multi-value Fields Within Periodic Groups* on page 2-67 for additional information.) Additionally, describing the same Adabas file in multiple ways enables you to access that same file in its different configurations within one procedure.

Master Files have three parts: file declaration, segment declaration, and field declaration, each of which is explained in these topics. Concepts that are specific to Adabas files, for which the server requires unique syntax in the Master File, are also explained.

The following is an example of the Master File for VEHICLES.

```
$$$ CREATED ON 12/10/97 AT 10.17.27 BY PMSMJB  
FILENAME=ADACAR,SUFFIX=ADBSINX,$
```

```
$ ADABAS FILE = VEHICLES-FILE          DICTIONARY = 6  
SEGNAME=S01      ,SEGTYPE=S,$  
  FIELD= REG_NUM          ,ALIAS= AA      ,A15  ,A15  ,INDEX=I,$  
  FIELD= CHASSIS_NUM      ,ALIAS= AB      ,I9    ,I4    ,  
  FIELD= PERSONNEL_ID     ,ALIAS= AC      ,A8    ,A8    ,INDEX=I,$  
  GROUP= CAR_DETAILS      ,ALIAS= CD      ,A50   ,A50   ,  
    FIELD= MAKE           ,ALIAS= AD      ,A20   ,A20   ,INDEX=I,$  
    FIELD= MODEL          ,ALIAS= AE      ,A20   ,A20   ,  
    FIELD= COLOR          ,ALIAS= AF      ,A10   ,A10   ,INDEX=I,$  
  FIELD= YEAR             ,ALIAS= AG      ,P2    ,Z2    ,  
  FIELD= CLASS            ,ALIAS= AH      ,A1    ,A1    ,INDEX=I,$  
  FIELD= LEASE_PUR        ,ALIAS= AI      ,A1    ,A1    ,  
  FIELD= DATE_ACQ         ,ALIAS= AJ      ,P6    ,Z6    ,  
$GRMU = CAR_MAINTENANCE ,ALIAS= AK      ,A11   ,A7    ,  
  FIELD= CURR_CODE        ,ALIAS= AL      ,A3    ,A3    ,  
  FIELD= MAINT_COST_CNT   ,ALIAS= AMC     ,I4    ,I2    ,  
  FIELD= DAT_ACQ_DESC     ,ALIAS= AN      ,A4    ,A4    ,INDEX=I,$  
  GROUP= MODEL_YEAR_MAKE ,ALIAS= AO      ,A28   ,A22   ,INDEX=I,$  
    FIELD=YEAR_S02        ,ALIAS= AG      ,P2    ,Z2    ,  
    FIELD=MAKE_S02        ,ALIAS= AD      ,A20   ,A20   ,INDEX=I,$  
  
SEGNAME=AM0101  ,SEGTYPE=S,PARENT=S01 ,OCCURS=AMC,$ MAX= 60  
  FIELD= MAINT_COST      ,ALIAS= AM      ,P7    ,P4    ,  
  FIELD= AM0101_OCC     ,ALIAS= ORDER   ,I4    ,I2    ,
```

File Attributes in Master Files

Master Files begin with a file declaration that names the file and describes the type of data source. The file declaration has two attributes, FILENAME and SUFFIX.

The syntax is

```
FILE[NAME]= filename, SUFFIX=ADBSINX [,,$]
```

where:

filename

Is a one- to eight-character name that complies with server file naming conventions.

ADBSINX

Is the value that specifies the adapter.

FILENAME

The FILENAME (or FILE) is any valid name from one to eight characters long.

In the OS/390 and z/OS environment, it is a member of a partitioned data set with the DDNAME MASTER.

SUFFIX

The value for the SUFFIX attribute is always ADBSINX, which is the name of the Data Adapter for Adabas program that the server loads to read an Adabas database.

Example File Declaration

The following example is a typical file declaration:

```
FILENAME=ADACAR, SUFFIX=ADBSINX, $
```

Segment Attributes in Master Files

Master Files begin with a file declaration that names the file and describes the type of data source. The file declaration has two attributes, FILENAME and SUFFIX.

Each segment declaration describes one Adabas record or a subset of an Adabas record (called a logical record type). Each logical record type described in a Master File requires a segment declaration that consists of at least two attributes, SEGNAME and SEGTYPE.

The syntax is

```
SEGNAME= segname, SEGTYPE=segtype [ , PARENT=parent ]
[ , OCCURS=occursname ] [ , $ ]
```

where:

segname

Is a unique one- to eight-character name that identifies the segment.

segtype

Identifies the characteristics of the segment. Possible values are:

S which indicates multiple instances of a descendant segment are related to a given parent.

SO which indicates the Adabas Write Data Adapter (for the Read Data Adapter, SEGTYPE=S is sufficient).

U which indicates a single instance of data for a descendant segment is related to its parent.

KL which indicates multiple instances of a cross-referenced descendant segment are related to a given parent.

KLU which indicates a single instance of a cross-referenced descendant segment is related to its parent.

parent

Is the name of the parent segment from the Adabas database.

occursname

Indicates a data field in the parent segment that contains the number of occurrences of the descendant segment. Its value is derived from the two-character internal Adabas name (ALIAS field on the server) appended with the letter C.

SEGNAME

The SEGNAME (or SEGMENT) attribute is the name of a group of data fields that have a one-to-one relationship to each other in the Adabas file.

You may describe an Adabas file in the server's hierarchical design to take advantage of multi-value fields and periodic groups. You may have multiple Adabas records described in the same server view. Each record is described as a segment with a unique SEGNAME.

Syntax

How to Name a Root Segment in the Master File

For the root segment of the Adabas file, the segment name is

Snn

where:

nn

Is a two-digit number indicating the order in which the file was selected.

The first file (selected as the root from the File Selection Menu) has SEGNAME=S01.

Subsequent files used in the same description (selected as children from the File Selection Menu) have SEGNAME=S02, SEGNAME=S03, and so on.

Syntax **How to Name a Segment for a PE Group and MU Field**

The segment names for PE (periodic element) groups and MU (multi-value) fields have the format

aammnn

where:

aa

Is the Adabas field name.

mm

Is a two-digit number that indicates the order in which the PE group or MU field appears in the PREDICT description of the file.

nn

Is the order number used for the root segment.

Example **Naming a Segment for a PE Group and MU Field**

Using the syntax *aammnn*, *SEGNAME=BE0201* describes the segment for the field BE, which is the second (02) PE group or MU field described in the first segment (01).

SEGTYPE

Is the root segment and cross-reference segment type. The SEGTYPE is S0 for the root file and children with a non-unique IXFLD. The SEGTYPE is U for a child with a unique IXFLD. The SEGTYPE for all PE and MU fields is S0.

PARENT

Is the value of SEGNAME of the parent record.

OCCURS

Indicates the number of occurrences of a PE group or MU field. This attribute contains the Adabas field name of the PE group or MU field with the suffix C, which is the ALIAS of the counter field in the parent segment.

SEGTYPE

The SEGTYPE attribute identifies the basic characteristics of related or cross-referenced segments. Cross-referencing is a method of accessing information from two or more different files or segments to use in a single report request.

PARENT

Any segment except the root is a descendant, or child, of another segment. The PARENT attribute supplies the name of the segment which is the logical parent or owner of the current segment. If no PARENT attribute appears, the default is the immediately preceding segment; however, it is highly recommended to include the parent. The PARENT name is the one- to eight-character SEGNAME of a previous segment.

OCCURS

For more information on the segment attribute OCCURS, which applies to repeating fields and groups, see *Describing Multi-value Fields and Periodic Groups With the OCCURS Attribute* on page 2-65.

Example Segment Declaration

The following two examples are typical segment declarations:

```
SEGNAME=S01          ,SEGTYPE=S          ,$  
SEGNAME=AM0101      ,SEGTYPE=S          ,PARENT=S01      ,OCCURS=AMC      ,$
```

Field Attributes in Master Files

Field declarations describe the fields in each file. For the simplest file structures (single-segment, fixed-length files), you need to describe only the Adabas fields you actually use for reporting purposes.

Put the fields in any order within their segment, except in the case of fields within periodic groups (PE). The PE group must have all fields defined and in the proper order.

The syntax is

```
FIELD[NAME]=fieldname , ALIAS=alias , [USAGE=] display , [ACTUAL=] format , $
```

where:

fieldname

Is the 1- to 66-character name that is unique to the Master File.

alias

Is the two-character internal Adabas field name.

display

Is the server display format for the field.

format

Is the server definition of the Adabas field format and length.

Note: Field declarations are always terminated with , \$.

There are additional attributes that apply to superdescriptor, cross-referenced, repeating, and other types of fields.

FIELD NAME

The field name appears as a column heading when you include the field in a report request. You can change the default by using AS or NOPRINT in the report or by using TITLE in the Master File.

Field names are unique names of 1-66 characters. The Master File field name does not need to be the same as the Adabas field name. Field names can include any alphanumeric characters, but the first character must be a letter from A to Z. Avoid embedded blanks and special characters.

Since all references to data on the server are through field names or aliases, the field names should be unique within a Master File. This requirement is true even when the same Adabas record is included as two differently structured segments in the same Master File. You must specify different field names in the two segments or the server uses the first one in the Master File unless you qualify which field you want, for example, file.field, segment.field.

ALIAS

The ALIAS for a single Adabas field *must* contain the two-character internal Adabas field name. The data adapter uses it to generate direct calls to the Adabas DBMS. The format rules are as follows:

- The ALIAS equals the two-character internal Adabas field name, for example, AM.
- The letter C is appended to the two-character internal Adabas field name if the field is the counter field for a PE group or MU field, for example, AMC.
- The ALIAS is the two-character internal Adabas field name appended with the digit 1 if an application calls for only the first occurrence of a repeating field, for example, AM1.
- The ALIAS can be used to reformat the length of the Adabas field. See *Using the ALIAS to Reformat Adabas Fields* on page 2-41 for examples using the reformatting option.

Note: Because of the requirement that the ALIAS in a Master File be the same as the two-character internal Adabas field name, your Master File may include duplicate ALIAS values. In that case, the field name is used to differentiate between the segments.

Using the ALIAS to Reformat Adabas Fields

The ALIAS field allows reformatting of the field for a different view. Several Adabas data formats have lengths that exceed the supported maximum length of data formats on the server. You can use the ALIAS to convert this data to an acceptable ACTUAL format and length.

When a field is specified in a retrieval request, the value in the ALIAS is passed to Adabas in the Format buffer. Adabas uses the ALIAS to process the field and return the data to the server.

To use the reformatting option, define the ALIAS using length and format values that indicate how you want to reformat the field.

The syntax is

```
ALIAS= 'xx,l,f'
```

where:

xx

Is the two-character internal Adabas field name.

l

Is the new length of the field value in number of bytes.

f

Is the new Adabas format of the field value.

The entire value must be enclosed in single quotation marks.

For example, if you want to use only the first three characters in a six-byte field, use the following syntax in your Master File:

```
FIELD=L_DBNR ,ALIAS='AU,3,P' , USAGE=P3, ACTUAL=P3 , $
```

In this case, the Data Adapter for Adabas will read only the first three bytes (L_DBNR is a six-byte field with the Adabas description: 01, AU, 6, U).

In the next example, to convert an Adabas binary field format of ten bytes (Adabas description: 01, BB, 10, B) to an alphanumeric format large enough to accommodate the Adabas value, you could use the following syntax in your Master File:

```
FIELD=TESTFLD ,ALIAS='BB,10,A' , USAGE=A20, ACTUAL=A10 , $
```

Notice that you must also change the USAGE and ACTUAL values to reflect the changes in the format. Refer to the chart in the *ACTUAL* on page 2-43 for the acceptable USAGE and ACTUAL values along with the corresponding Adabas codes.

Note: Groups cannot be reformatted. This formatting is valid only for elementary fields.

USAGE

The USAGE attribute enables you to describe how you want fields displayed on the screen, printed in reports, or used in calculations. This attribute includes the data field type, length, and any applicable edit options when the field values are printed.

The syntax is

```
{USAGE}=t111e
```

where:

USAGE

Indicates how to display a field. FORMAT is a synonym for USAGE.

t

Is the field type.

111

Is the number of positions needed to display the field.

e

Are the edit options.

The value that you specify for field length is the number of alphanumeric characters or numeric digits that the server allocates for the field in any display or report. The specified value excludes the editing characters, such as comma, \$, and so on. Edit options control how a field value is printed.

ACTUAL

The ACTUAL attribute describes the length and type of the Adabas field in storage. It is used to describe the format of fields from external data files only, and must be included in the Master Files needed for the Data Adapter for Adabas. The source of this information is your existing Adabas Field Definition Table (FDT).

The syntax is

```
ACTUAL=t111
```

where:

t

Is the server field type of the Adabas field.

111

Is the storage length of the Adabas field.

The server permits the following conversions from ACTUAL format to USAGE (display) format:

Adabas Format Type	Server ACTUAL Type	Server ACTUAL Length*	Server USAGE Type	Server USAGE Length*	Description
A	A	1-256	A	1-256	Alphanumeric
G	F D	4 8	F D	1-9 1-15	Float Single-Precision Float Double-Precision
F	I	2, 4	I	1-9	Integer
B 1-4 5-6 7-126	I P A	1-4 7-8 9-126	I P A	1-9 12-15 14-252	Integer Values Packed Decimal Alpha
P	P	1-15	P	1-33. <i>n</i>	Packed Decimal
U	Z	1-29	A or P	1-33. <i>n</i>	Zoned

* Lengths are measured as follows:

- ACTUAL is in number of bytes.
- USAGE allows space for all possible digits, a minus sign for negative numbers, and a decimal point in numbers with decimal digits.

The following table shows the codes for the types of data the server reads:

ACTUAL Type	Description
<i>An</i>	Alphanumeric characters A to Z, 0 to 9, and the special characters in the EBCDIC display mode, where <i>n</i> = 1 to 256 bytes.
<i>D8</i>	Double-precision, floating-point numbers, stored internally in eight bytes.
<i>F4</i>	Single-precision, floating-point numbers, stored internally in four bytes.
<i>In</i>	Binary integers, where <i>n</i> = 1 to 4.
<i>Pn</i>	Packed decimal data, where <i>n</i> = 1 to 15 bytes.

ACTUAL Type	Description
<i>Zn</i>	Zoned decimal data, where $n = 1$ to 30 bytes. Represent the field as $Zm.n$, where m is the total number of digits, and n is the number of decimal places (for example, Z6.1 means a six-digit number with one decimal place).

INDEX

For more information on the field attribute INDEX, which applies to descriptors, see *Specifying INDEX=I* on page 2-54.

GROUP

For more information on the field attribute GROUP, which applies to superdescriptors, see *Specifying Superdescriptors Using the GROUP Attribute* on page 2-55.

Example Field Declaration

The following is a typical field declaration:

```
FIELD=MODEL ,ALIAS=AE ,USAGE=A20 ,ACTUAL=A20 ,,$
```

Access Files for Adabas

An Access File describes the physical attributes of the Adabas file, such as the database number, file number, descriptors, and security. See *Overview of Master and Access Files* on page 2-35 for more information about Master and Access Files.

Access Files store database access information used to translate report requests into the required Adabas direct calls.

In the OS/390 and z/OS environment, the Access File is located in a PDS allocated to the DDNAME ACCESS. Each member is a separate Access File.

An Access File consists of 80-character records in comma-delimited format. A record in an Access File contains a list of attributes and values, separated by commas and terminated with a comma and dollar sign (,\$). This list is free-form and spans several lines if necessary. You can specify attributes in any order.

The server reads blank lines, and lines starting with a dollar sign in column one, as comments.

Every Access File contains a release declaration and at least one segment declaration. Release declarations and segment declarations each have their own set of attributes. These attributes are discussed in the following topics.

The following is a sample Access File:

```
$$$ FILENAME=EMPFIL1,SUFFIX=ADBSINX,$
RELEASE=6,OPEN=YES,$

$ ADABAS FILE = EMPLOYEES                                DICTIONARY =
SEGNAM=S01      ,ACCESS=ADBS,FILENO=001,DBNO=1,CALLTYPE=RL,
                SEQFIELD=PERSONNEL_ID,$
FIELD= DEPT_PERSON                                     ,TYPE=SPR,$
FIELD=DEPT                                             ,TYPE=DSC,NU=YES,$
FIELD=NAME                                             ,TYPE=    ,NU=NO,$
FIELD= LEAVE_LEFT                                     ,TYPE=SPR,$
FIELD=LEAVE_DUE                                       ,TYPE=    ,NU=YES,$
FIELD=LEAVE_TAKEN                                    ,TYPE=    ,NU=YES,$
FIELD= DEPARTMENT                                    ,TYPE=NOP,NU=NO,$
SEGNAM=AI0101,ACCESS=MU    ,FILENO=001,$ ADDRESS_LINE
SEGNAM=AQ0201,ACCESS=PE    ,FILENO=001,DBNO=1,$ INCOME
SEGNAM=AT0301,ACCESS=MU    ,FILENO=001,DBNO=1,$ BONUS
SEGNAM=AW0401,ACCESS=PE    ,FILENO=001,$ LEAVE_BOOKED
SEGNAM=AZ0501,ACCESS=MU    ,FILENO=001,$ LANG
```

Release Declaration

The release declaration must be the first uncommented line of the Access File. It identifies the release of Adabas and indicates whether the Data Adapter for Adabas issues Adabas OPEN and CLOSE calls for each report request.

The syntax is

```
RELEASE= relnum [,OPEN={YES|NO}] , $
```

where:

relnum

Is the Adabas release number.

OPEN

Specifies whether the adapter issues Adabas OPEN and CLOSE calls to Adabas in each report request. Possible values are:

YES which indicates that Adabas OPEN and CLOSE calls are issued. This value is the default.

NO which indicates that Adabas OPEN and CLOSE calls are not issued.

RELEASE

The first declaration in the Access File contains the Adabas release number with the attribute RELEASE. Specify the full release number. The value is for documentation only.

The following example illustrates the use of the RELEASE attribute to specify the Adabas release number:

```
RELEASE=6 , $
```

OPEN

The OPEN attribute specifies whether the Data Adapter for Adabas issues Adabas OPEN and CLOSE calls to Adabas in each report request. Specifying YES or NO achieves the following results:

Value	Result
YES	The adapter issues an OPEN call to Adabas at the start of a call set initiated by a retrieval request. An Adabas CLOSE is issued at the end of retrieval for the request. YES is the default value. <code>RELEASE=6 , OPEN=YES , \$</code>
NO	Adabas OPEN and CLOSE calls are not issued for any retrieval request. <code>RELEASE=4.1 , OPEN=NO , \$</code>

Accept the default value (YES) unless you are using an Adabas release lower than 5.0.

Segment Attributes in Access Files

Segment declarations contain detailed information about each segment of an Adabas file. The segment declaration attributes specify the segment name, the file and database numbers, and the Adabas password, as well as retrieval options.

The syntax is

```
SEGNAM=segname , ACCESS=access , FILENO=file_number [ ,DBNO=database_number ]
[ ,CALLTYPE={FIND|RL} ] [ ,SEQFIELD=seqfield ] [ ,PASS=password ]
[ ,FETCH = {ON|OFF} ] [ ,FETCHSIZE = {n|MAX} ] [ ,UNQKEYNAME=unique_field ]
[ ,WRITE=YES|NO ] , $
```

where:

segname

Is the SEGNAME value from the Master File.

access

Specifies the segment type that the adapter uses for each segment. Possible values are: **ADBS** which is for a segment that contains only non-repeating fields. This value is the default.

PE which is for a segment that is a periodic group in the same file as the parent segment.

MU which is for a segment that is a multi-value field in the same file as the parent segment.

file_number

Is the number that identifies an Adabas file.

database_number

Is the number that identifies an Adabas database.

CALLTYPE

Indicates the type of data retrieval call constructed by the adapter. Possible values are: **FIND** in which a FIND call is issued when there are one or more WHERE or IF statements in a retrieval request detected against fields defined with INDEX=I. This is the default value.

RL in which a Read Logical (RL) call is issued when there are one or more WHERE or IF statements against fields defined with INDEX=I.

Note: See your database administrator for the most efficient CALLTYPE to be used at your site.

seqfield

Provides a default index which controls Read Logical (RL) retrieval when there is no IF or WHERE selection test.

password

Is the Adabas password for the file.

FETCH

Indicates whether to use the Fetch feature. Possible values are:

ON which sets the Fetch feature on for the user session. This is the default value.

OFF which sets the Fetch feature off for the user session.

If you include this attribute, you must also include FETCHSIZE (see below).

FETCHSIZE

Sets the number of records per buffer. Possible values are:

n which is the number of records per buffer (1 to 999). Ten records per buffer is the default.

MAX which is automatically calculated by the adapter to allow the maximum number of records that fit in a 32K buffer.

UNQKEYNAME which is the attribute that indicates the unique key. It does not necessarily define key described in the ADABAS FDT with the UQ option.

WRITE which is the attribute that indicates the WRITE capability.

Note: FETCH and FETCHSIZE are applicable only to segments described as ACCESS=ADBS. FETCH and FETCHSIZE can be set dynamically to override the Access File settings.

Special attributes that apply to embedded JOINS are explained in *Implementing Embedded JOINS: KEYFLD and IXFLD* on page 2-59. Attributes that apply to superdescriptors and subdescriptors are explained in *Customizing the Adabas Environment* on page 2-72.

SEGNAM

The SEGNAM attribute specifies the name of the segment being described. It is the same name as the SEGMENT value in the Master File.

ACCESS

The ACCESS attribute specifies the segment type that the Data Adapter for Adabas uses for each segment. The values are:

- ADBS, which specifies a segment that contains only non-repeating fields. This value is the default.
- PE, which specifies a segment that is a periodic group in the same file as the parent segment.
- MU, which specifies a segment that is a multi-value field in the same file as the parent segment.

All three access values use the attributes FILENO and DBNO, as shown in this partial Access File for EMPLOYEES with the specification of PE and MU fields.

```
$$$ FILENAME=EMPFIL1, SUFFIX=ADBSINX, $
RELEASE=6, OPEN=YES, $
```

```
$ ADABAS FILE = EMPLOYEES                                DICTIONARY =
SEGNAM=S01      ,ACCESS=ADBS, FILENO=001, DBNO=1, CALLTYPE=RL,
                SEQFIELD=PERSONNEL_ID, $
SEGNAM=AQ0201, ACCESS=PE    , FILENO=001, DBNO=1, $ INCOME
SEGNAM=AT0301, ACCESS=MU    , FILENO=001, DBNO=1, $ BONUS
SEGNAM=AW0401, ACCESS=PE    , FILENO=001, DBNO=1, $ LEAVE_BOOKED
```

ACCESS=ADBS

ACCESS=ADBS specifies the main segment of the file. This segment contains only non-repeating fields.

FILENO

The FILENO attribute specifies the Adabas file number. You need this file number to identify the Adabas file(s) you wish to access. The valid values are 1 to 255 (5000 is the maximum value for a mainframe platform). The FILENO attribute has to be defined for segments with ACCESS=ADBS. If the FILENO attribute is omitted for segments with ACCESS=MU/PE, then it will be taken from the corresponding parent segment.

Suppose you are accessing two segments: the first is in the EMPLOYEE file, and the second is in the INSURANCE file. To describe this situation in the Access File, use the SEGNAME and FILENO attributes as illustrated in DBNO.

DBNO

The DBNO attribute specifies the Adabas database number. It is used when you define files from multiple databases in one Master File. If it is not provided, the DBNO will be read from the DDCARD (on a mainframe platform only). If the DDCARD is not allocated, the data adapter uses the default value, DBNO=0.

If the DBNO attribute is omitted for a segment with ACCESS=PE/MU, then it will be taken from the corresponding parent segment. The DBNO attribute has to be defined for segments with ACCESS=ADBS on non-mainframe platforms.

The following example illustrates the use of both the FILENO and DBNO attributes to account for the files being in different databases:

```
SEGNAME=ONE      ,FILENO=1  ,DBNO=1  ,...  ,$
```

```
SEGNAME=PAY      ,FILENO=2  ,DBNO=3  ,...  ,$
```

CALLTYPE

The CALLTYPE attribute affects how the Data Adapter for Adabas processes WHERE and IF statements on descriptors in report requests and how it retrieves cross-referenced file data from a descendant segment. With CALLTYPE, you optionally specify the type of access to use to process the inverted lists for any given segment. Ask your Adabas database administrator to advise you when you are selecting a CALLTYPE.

The possible values and their meanings are:

Value	Description
FIND	<p>An Adabas FIND call is issued when there are one or more WHERE or IF statements against fields defined to the server with INDEX=I. The search is on specific values of the field.</p> <p>The Data Adapter for Adabas can generate Adabas FIND (S1) calls even when non-descriptor fields are used as search criteria. This can occur whenever CALLTYPE=FIND is specified in the Access File, and if you include an IF or WHERE test when referencing a non-descriptor field in your TABLE or TABLEF request.</p> <p>To turn off this feature, see the topic in <i>Data Adapter Navigation</i> on page 2-94 that discusses the optimization of the FIND call using non-descriptor fields.</p>

Value	Description
RL	<p>An Adabas Read Logical call is issued when there are one or more WHERE or IF statements against fields defined to the server with INDEX=I.</p> <p>Note: If you have selected a field defined with TYPE=NOP (for example, subdescriptors), a FIND call is issued even if RL has been specified in cases when:</p> <ul style="list-style-type: none"> • SET NOPACCESS FIND was performed. • The server is running in OpenVMS environment. • The same Adabas field is defined in the descendant PE/MU segment. <p>Switching from RL to FIND mode is also performed if:</p> <ul style="list-style-type: none"> • A field is defined with TYPE=SPR and the server is running in an OpenVMS environment. • A field is defined with TYPE=PDS (phonetic descriptor) or TYPE=HDS (hyper descriptor). • A field is defined in a PE segment (is a part of periodic group). <p>The Adabas Interface uses Read Logical by ISN (L1) calls to retrieve all records for the entry segment when:</p> <ul style="list-style-type: none"> • The report request does not contain an optimizable selection test on an inverted list or an ISN list. • The Access File contains a SEQFIELD value for that segment and this value is equal to the ISN field name. <p>Adabas returns each record corresponding to an entry in the Address Converter. The records are in ascending value of the ISN.</p> <p>The first IF statement in a request that refers to a field with INDEX=I determines the inverted list used for the Read Logical access for the root of the subtree. Refer to <i>Referencing Subtrees and Record Retrieval</i> on page 2-78 for information on subtrees.</p> <p>When retrieving descendant segments, the Data Adapter for Adabas issues a Read Logical call using the descriptor pointed to by IXFLD. For more information on IXFLD, refer to <i>Implementing Embedded JOINS: KEYFLD and IXFLD</i> on page 2-59.</p> <p>For CALLTYPE=RL, all other selection tests on descriptor fields specified in your report request are applied after the record is returned.</p>

SEQFIELD

The SEQFIELD attribute specifies the field you want to use as the sequencing value:

```
RELEASE=6 , OPEN=YES , $
```

```
$ ADABAS FILE = EMPLOYEES                                DICTIONARY =
SEGNAM=S01      , ACCESS=ADBS , FILENO=001 , DBNO=1 , CALLTYPE=RL ,
                  SEQFIELD=PERSONNEL_ID , $
```

PASS

The PASS attribute specifies the Adabas password for the file. It is required if the file is password protected.

You must specify the password for each password-protected file you use. For example:

```
SEGNAM=ONE      , FILENO=1      , DBNO=1      , PASS=ONEXX , . . . , $
SEGNAM=PAY      , FILENO=2      , DBNO=3      , PASS=USER2  , . . . , $
```

FETCH

The FETCH attribute indicates whether to use the Fetch feature for segments described as ACCESS=ADBS in the Access File. Valid values are ON (default) or OFF.

FETCHSIZE

The FETCHSIZE attribute sets the number of records per buffer. You can supply a specific number or have the Data Adapter for Adabas automatically calculate the maximum number of records that fit in a 32K buffer.

The following is an example of an Access File that enables the Fetch feature and sets the number of records per buffer to 15.

```
$$$ FILENAME=EMPFILE1 , SUFFIX=ADBSINX , $
RELEASE=6 , OPEN=YES , $
```

```
$ ADABAS FILE = EMPLOYEES                                DICTIONARY =
SEGNAM=S01      , ACCESS=ADBS , FILENO=001 , CALLTYPE=RL ,
                  SEQFIELD=PERSONNEL_ID , FETCH=ON , FETCHSIZE=15 , $
```

Mapping Adabas Descriptors

Adabas descriptors, superdescriptors, and subdescriptors must be declared in Master and Access Files in certain ways:

- When creating Master Files for fields that are descriptors, use the INDEX=I attribute. The Access File must also contain information about the descriptors, specifying the TYPE as DSC, SPR, or NOP. Identifying descriptors in the Access File directs the server to inverted lists to speed retrieval of these fields.
- Adabas fields described as superdescriptors must be declared in the Master File using the GROUP attribute. In the Access File, the TYPE must be SPR.

Specifying INDEX=I

Descriptors act as key values associated with a single field in an Adabas record. They enable records to be retrieved more efficiently based on the selection of a value, a set of values, or a range of values.

Additionally, all descriptors, superdescriptors, and subdescriptors are used to establish logical relationships between files. There are two methods for establishing logical relationships:

- By establishing the relationship dynamically at runtime with the JOIN command. Refer to *Data Adapter Navigation* on page 2-94 for more information about using Adabas files that are to be cross-referenced using JOIN.
- By using a static cross-reference in the Master and Access Files. See *Implementing Embedded JOINS: KEYFLD and IXFLD* on page 2-59 for information on how to define shared files.

Adabas descriptor fields are identified in the Master File using the INDEX=I attribute. For example:

```
FIELD=MAKE ,ALIAS=AD ,USAGE=A20 ,ACTUAL=A20 ,INDEX=I, $
```

INDEX=I is optional in the Master File, but using it is recommended so that keys in your Master File are easily recognized without reference to the Access File.

Describing Superdescriptors

A superdescriptor is a key value associated with all or part of two to twenty Adabas fields. If the superdescriptor consists of a group of complete fields, it is declared differently than a superdescriptor which consists of one or more partial fields. For information on superdescriptors consisting of partial fields, see *Customizing the Adabas Environment* on page 2-72.

Superdescriptors can be printed if they contain only alphabetic fields.

Specifying Superdescriptors Using the GROUP Attribute

A superdescriptor composed of several complete fields is described with the GROUP attribute. All the fields that are part of the group must be specified immediately after the GROUP attribute in the order in which they appear in the superdescriptor.

The following example shows a superdescriptor defined in server terminology:

```
GROUP=TRANS           ,ALIAS=S1   ,USAGE=A20   ,ACTUAL=A20   ,INDEX=I   , $
  FIELD=ACCT_NO       ,ALIAS=C2   ,USAGE=A9    ,ACTUAL=A9    , $
  FIELD=ITEM_NO        ,ALIAS=GD   ,USAGE=A5    ,ACTUAL=A5    , $
  FIELD=TRANS_DATE    ,ALIAS=D7   ,USAGE=A6YMD ,ACTUAL=A6    , $
```

The group field name, in this case, is TRANS. Since it is a superdescriptor, the INDEX=I attribute is included on the GROUP specification.

The ALIAS is the two-character Adabas field name for the superdescriptor.

The USAGE and ACTUAL values must be specified in alphanumeric format. See the next topic on calculating group length for more information.

If any of the components of the TRANS group field were descriptors, they could also contain the INDEX=I attribute.

Note: TYPE=SPR must be defined in the Access File. For more information, see *Customizing the Adabas Environment* on page 2-72.

Calculating GROUP Length

For a group field, you must supply both the USAGE and ACTUAL values in alphanumeric format.

If the component fields of the group are alphanumeric, the USAGE and ACTUAL lengths of the group field must be exactly the sum of the component field lengths. For example, in the *Specifying Superdescriptors Using the GROUP Attribute* on page 2-55, the lengths specified in the USAGE and ACTUAL attributes for the component fields ACCT_NO, ITEM_NO, and TRANS_DATE each total 20. The lengths specified in the USAGE and ACTUAL attributes of the TRANS group field are consequently both equal to 20.

If the component fields of the group are *not* alphanumeric:

- The USAGE and ACTUAL formats of the group field must still be alphanumeric.
- The ACTUAL length of the group field is still the sum of the component field lengths.
- The USAGE length is the sum of the internal storage lengths of the component fields.

You determine these internal storage lengths as follows:

USAGE	Length
A (alphanumeric)	A value equal to the number of characters contained in the field
D (decimal, double-precision)	8
F (decimal, single precision)	4
I (integer)	4
P (packed decimal):	
Pn or Pn.d, where n is less than or equal to 15	8
P16	16
P16.d	8
Pn or Pn.d, where n is greater than 16 or less than or equal to 31	16
Z (zoned)	A value equal to the number of characters contained in the field

For example, consider the following GROUP specification in the EMPLOYEES Master File:

```
GROUP= LEAVE_DATA           ,ALIAS= A3           ,A16   ,A4   ,$
      FIELD=LEAVE_DUE       ,ALIAS= AU          ,P2    ,Z2   ,$
      FIELD=LEAVE_TAKEN     ,ALIAS= AV          ,P2    ,Z2   ,$
```

In this example:

- The lengths of the ACTUAL values for the component fields LEAVE_DUE and LEAVE_TAKEN total 4, which is the length of the ACTUAL value for the group field.
- The length of the USAGE value for the group field is determined by adding the internal storage lengths of the component fields LEAVE_DUE and LEAVE_TAKEN, as specified by the field types: a length of 8 for USAGE P2 (8 plus 8 = 16).

Here is a GROUP specification:

```
GROUP= MODEL_YEAR_MAKE     ,ALIAS= AO          ,A28   ,A22   ,INDEX=I,$
      FIELD=YEAR            ,ALIAS= AG          ,P2    ,Z2    ,$
      FIELD=MAKE            ,ALIAS= AD          ,A20   ,A20   ,INDEX=I,$
```


In this example:

- The lengths of the ACTUAL values for the component fields YEAR and MAKE total 22, which is the length for the ACTUAL value for the group field.
- The length of the USAGE value for the group field is determined by adding the internal storage lengths of the component fields YEAR and MAKE, as specified by the field types: a length of 8 for USAGE P2 and a length of 20 for USAGE A20 (8 plus 20 = 28).

Describing Descriptors in the Access File

Field suffixes are specified in the Access File to identify descriptors, superdescriptors, and subdescriptors.

The syntax is

```
FIELD[NAME]=fieldname, TYPE=fieldsuffix, [NU={YES|NO}] , $
```

where:

fieldname

Is the field name or group in the Access File.

fieldsuffix

Indicates the field suffix. Possible values are:

DSC which indicates a descriptor.

SPR which indicates a superdescriptor made up of whole fields.

NOB which indicates a subdescriptor or superdescriptor made up of partial fields.

PDS which indicates a phonetic descriptor.

HDS which indicates a hyper descriptor.

blank which indicates non-descriptor fields.

NU

Specifies whether null-suppression is in use. Possible values are:

YES which indicates that the field is described in the data adapter with null-suppression.

Note: Descriptors with null values are not stored in inverted lists.

NO which indicates null-suppression is not used. This is the default value.

Specifying Superdescriptors Containing Partial Fields

If a superdescriptor consists of one or more partial fields, that superdescriptor (field) is defined with TYPE=NOP (non-printable) in the Access File. This type of descriptor can be used in selection tests, but neither it nor any part of it can be printed or displayed unless the field is alphanumeric.

The partial fields that comprise such superdescriptors are stored in Adabas; there is no facility or necessity for identifying them to the server or the data adapter. If you look at an Access File that contains a superdescriptor composed of partial fields, you would recognize it by TYPE=NOP, but you cannot list the partial fields that are its components. To use the adapter, identify such a superdescriptor in the Access File with TYPE=NOP.

Any superdescriptor defined to the server with TYPE=NOP has the following limitation: CALLTYPE=RL is not supported when this field is the IXFLD in an embedded cross-reference or JOIN.

See your Software AG documentation for more information about using superdescriptors containing partial fields.

Specifying Subdescriptors

A subdescriptor consists of a partial field value for which an index has been created in Adabas. It is described at the field level in the Access File as TYPE=NOP. It can be used in selection tests, but it cannot be printed.

Any subdescriptor defined to the server in the Access File with TYPE=NOP has the following limitation: CALLTYPE=RL is not supported when this field is the IXFLD in an embedded cross-reference or JOIN.

Specifying Phonetic Descriptors

A phonetic descriptor consists of similar phonetic values for which an index has been created in Adabas. It is described at the field level in the Access File as TYPE=PDS. It can be used in selection tests, but it cannot be printed.

Specifying Hyper Descriptors

A hyper descriptor consists of values generated, based on a user-supplied algorithm, for which an index has been created in Adabas. It is described at the field level in the Access File as TYPE=HDS. It can be used in selection tests, but it cannot be printed.

Any phonetic/hyper descriptor defined to the server in the Access File with TYPE=PDS/HDS has the following limitation: CALLTYPE=RL is not supported when this field is the IXFLD in an embedded cross-reference or JOIN.

If you have selected a field defined with TYPE=NOP (for example, subdescriptors), TYPE=PDS or TYPE=HDS, a FIND call is issued even if RL has been specified.

Specifying Null-Suppression

To allow the Data Adapter for Adabas to create the most efficient calls, while still maintaining integrity of the answer set, any null-suppressed fields that are components of a superdescriptor must be defined in the Access File. The NU attribute is used to define a null-suppressed field.

An NU field defined as a descriptor is not stored in inverted lists when it contains a null value. Any qualifying descriptor records containing a null value are not recognized by a FIND command that refers to that descriptor.

The same is true for subdescriptors and superdescriptors derived from fields described in the data adapter with null-suppression. No entry is made for a subdescriptor if the bytes of the field from which it is derived contain a null value and that field is defined with null-suppression (NU). No entry is made for a superdescriptor if any of the fields from which it is derived is an NU field with a null value.

For more information about null-suppression and how it affects data retrieval, see your Software AG documentation.

Implementing Embedded JOINS: KEYFLD and IXFLD

The KEYFLD and IXFLD attributes identify the common fields for parent/descendant relationships in a multi-segment Master File. These relationships are referred to as embedded JOINS or server views.

For each descendant segment, the KEYFLD and IXFLD attributes specify the field names of the shared field that implements the embedded JOIN. The parent field supplies the value for cross-referencing; the descendant field contains the corresponding value. The data adapter implements the relationship by matching values at run time.

The KEYFLD and IXFLD attributes are valid only for ACCESS=ADBS segments.

Attribute	Description
KEYFLD	Refers to the field name in the parent segment whose value is used to retrieve the child segment. This is also known as the primary key.
IXFLD	Refers to the field name in the child or cross-referenced segment containing the related data. This is also known as the foreign key.

The value for the KEYFLD attribute is a 1- to 66-character field name or alias from the parent segment. The value for the IXFLD attribute is a 1- to 66-character field name or alias from the descendant segment. This is an example of a Master File with embedded JOIN using KEYFLD and IXFLD.

```

FILENAME=AMKTORDR      , SUFFIX=ADBSINX , $
SEGNAME=MKTORDER
FIELDNAME  =NMARKET_GRP      , BA      , I3      , I2,      INDEX=I , $
FIELDNAME  =QPRODUCT         , BB      , I3      , I2      , $
FIELDNAME  =QNEEDITM         , BC      , A3      , I2      , $
FIELDNAME  =FBUILD           , BD      , A1      , A1      , $
FIELDNAME  =FK_NPRODUCT      , BE      , A4      , A4      , $
FIELDNAME  =FK_NCUSTOMER     , BF      , I3      , I2      , $
FIELDNAME  =DATEMKTO         , BG      , A8      , A8      , $
FIELDNAME  =DATEFBLD         , BH      , A8      , A8      , $

SEGNAME=CUSTOMER , SEGTYPE=U , PARENT=MKTORDER , $
FIELDNAME  =NCUSTMR_GRP      , AA      , I3      , I2,      INDEX=I , $
FIELDNAME  =NAMECUST         , AB      , A15     , A15     , $
FIELDNAME  =DCUSROAD         , AC      , A20     , A20     , $
FIELDNAME  =DCUSTOWN         , AD      , A20     , A20     , $

```

This is an example of an Access File with embedded JOIN using KEYFLD and IXFLD.

```

RELEASE=6 , OPEN=YES , $
  SEGNAM=MKTORDER , ACCESS=ADBS , FILENO=022 , DBNO=001 , $
  SEGNAM=CUSTOMER , ACCESS=ADBS , FILENO=021 , DBNO=001 , CALLTYPE=FIND ,
IXFLD=NCUSTMR_GRP , KEYFLD=FK_NCUSTOMER , $

```

Note: Include the pair of attributes in the Access File segment declaration for descendant segments. Do not specify them in the segment declaration for the root segment.

A JOIN can be based on more than one field in the host and cross-referenced logical record types. If the Adabas file uses multiple fields to establish a relationship or link between logical record types, you can specify concatenated fields in an embedded JOIN. You can also specify multiple fields with the dynamic JOIN command.

In the multi-field embedded JOIN, the KEYFLD and IXFLD values consist of a list of their component fields separated by slashes (/). Additional Access File attributes are not required. The syntax is

```

KEYFLD = field1/field2/... ,
IXFLD  = cfield1/cfield2/... , $

```

where:

```
field1/...
```

Is a composite of up to 16 key fields from the parent segment. Slashes are required.

cfield1/...

Is a composite of up to 16 key fields from the descendant segment.

The data adapter compares each field pair for the data formats prior to format conversion; it evaluates each field pair with the following rules:

- The cross-referenced field in the embedded JOIN must be an Adabas descriptor field. The host field can be any field.
- Parent and descendant fields must be real fields.
- All participating fields for either the parent or descendant file must reside in the same segment.
- KEYFLD and IXFLD attributes must specify the same number of fields. If the format of the parent field is longer than the format of the descendant field, its values are truncated. If the format of the parent field is shorter, its values are padded with zeros or blanks.
- The KEYFLD and IXFLD attributes can consist of a maximum of 16 concatenated fields in order of high-to-low significance. You must specify the field order: the order determines how the values will be matched.
- The pair of fields in the KEYFLD/IXFLD attribute does not have to have comparable data formats.

To implement JOINS, the Adabas DBMS converts the alphanumeric field formats on the server to equivalent Adabas field formats in order to perform the necessary search and match operations. When the Adabas DBMS returns the answer set of matched values, it also converts the values back to alphanumeric formats. The cross-referenced fields must be descriptor fields.

Mapping Adabas Files With Variable-length Records and Repeating Fields

The OCCURS segment attribute is used by the server to describe MU and PE field types in a Master File.

Consider the sample Adabas file VEHICLES. The FDT for the VEHICLES file is shown below.

```
*****
* FILE      2 (VEHICLES      ) *
*****
FIELD DESCRIPTION TABLE
```

LEVEL	I	I	I	I	I	I	I	I	I	
	I	NAME	I	LENGTH	I	FORMAT	I	OPTIONS	I	PARENT OF
1	I	AA	I	15	I	A	I	DE,UQ,NU	I	
1	I	AB	I	4	I	F	I	FI	I	
1	I	AC	I	8	I	A	I	DE	I	
1	I	CD	I		I		I		I	
2	I	AD	I	20	I	A	I	DE,NU	I	SUPERDE
2	I	AE	I	20	I	A	I	NU	I	
2	I	AF	I	10	I	A	I	DE,NU	I	
1	I	AG	I	2	I	U	I	NU	I	SUPERDE
1	I	AH	I	1	I	A	I	DE,FI	I	
1	I	AI	I	1	I	A	I	FI	I	
1	I	AJ	I	6	I	U	I	NU	I	SUPERDE
1	I	AK	I		I		I		I	
2	I	AL	I	3	I	A	I	NU	I	
2	I	AM	I	4	I	P	I	MU ,NU	I	

```
-----
SPECIAL DESCRIPTOR TABLE
```

TYPE	I	I	I	I	I	I	I	I	I	
	I	NAME	I	LENGTH	I	FORMAT	I	OPTIONS	I	STRUCTURE
SUPER	I	AN	I	4	I	B	I	DE,NU	I	AJ (5 - 6)
SUPER	I	AO	I	22	I	A	I	DE,NU	I	AG (1 - 2)
	I		I		I		I		I	AD (1 - 20)

```
-----
```

The MU field, MAINT_COST, is a field in VEHICLES. It is allocated to a new segment, **AM0101**, required by the Data Adapter, as shown in the Master File for a partial view of the VEHICLES file, illustrating the use of OCCURS=occursname:

```

$$$ CREATED ON 12/10/97 AT 10.17.27 BY PMSMJ
FILENAME=ADACAR,SUFFIX=ADBSINX,$

$ ADABAS FILE = VEHICLES-FILE          DICTIONARY = 6
SEGNAME=S01      ,SEGTYPE=S,$

FIELD= REG_NUM                ,ALIAS= AA      ,A15 ,A15 ,INDEX=I,$
FIELD= CHASSIS_NUM            ,ALIAS= AB      ,I9  ,I4  ,,$
FIELD= PERSONNEL_ID           ,ALIAS= AC      ,A8  ,A8  ,INDEX=I,$
FIELD= CLASS                   ,ALIAS= AH      ,A1  ,A1  ,INDEX=I,$
FIELD= LEASE_PUR               ,ALIAS= AI      ,A1  ,A1  ,,$
FIELD= DATE_ACQ               ,ALIAS= AJ      ,P6  ,Z6  ,,$
$GRMU = CAR_MAINTENANCE       ,ALIAS= AK      ,A11 ,A7  ,,$
FIELD= CURR_CODE              ,ALIAS= AL      ,A3  ,A3  ,,$
FIELD= MAINT_COST_CNT         ,ALIAS= AMC     ,I4  ,I2  ,,$
FIELD= DAT_ACQ_DESC           ,ALIAS= AN      ,A4  ,A4  ,INDEX=I,$
GROUP= MODEL_YEAR_MAKE       ,ALIAS= AO      ,A28 ,A22 ,INDEX=I,$
FIELD=YEAR_S02                ,ALIAS= AG      ,P2  ,Z2  ,,$
FIELD=MAKE_S02                ,ALIAS= AD      ,A20 ,A20 ,INDEX=I,$

SEGNAME=AM0101 ,SEGTYPE=S,PARENT=S01 ,OCCURS=AMC,$ MAX= 60
FIELD= MAINT_COST                ,ALIAS= AM      ,P7  ,P4  ,,$
FIELD= AM0101_OCC                ,ALIAS= ORDER   ,I4  ,I2  ,,$

```

In the original segment, the MAINT_COST_CNT (ALIAS=AMC) field contains the total number of occurrences of the MAINT_COST field. The new segment, AM0101, contains the MAINT_COST data field and the newly created AM0101_OCC field. The AM0101_OCC field contains the position of each occurrence.

This example includes the group field called CAR_MAINTENANCE, which contains the MU field, MAINT_COST. You cannot report on a group with an unknown length. However, the component fields can be referenced for reporting purposes.

Note: The CREATE SYNONYM facility creates the new segment, AM0101, for you. Otherwise, you must create the new segment using the OCCURS segment attribute.

The following is the Access File for the VEHICLES file:

```

$$$ CREATED ON 12/10/97 AT 10.17.27 BY PMSMJ
$$$ FILENAME=ADACAR , SUFFIX=ADBSINX , $
RELEASE=6 , OPEN=YES , $

$ ADABAS FILE = VEHICLES-FILE           DICTIONARY = 6
SEGNAM=S01 , ACCESS=ADBS , FILENO=002 ,
      CALLTYPE=RL , SEQFIELD=PERSONNEL_ID , $
FIELD= DAT_ACQ_DESC                       , TYPE=NOP , $
FIELD= MODEL_YEAR_MAKE                   , TYPE=SPR , $
FIELD=YEAR_S02                           , TYPE= , NU=YES , $
FIELD=MAKE_S02                           , TYPE=DSC , NU=YES , $
SEGNAM=AM0101 , ACCESS=MU , FILENO=002 , $ MAINT_COST

```

A new segment is created for the MU field. Note that for this new segment, ACCESS=MU.

Here is an example of the counter fields. The AMC field contains the total count of occurrences, the AM field contains the actual data, and the ORDER field contains the position of each occurrence.

(AMC)	(AM)	(ORDER)
MAINT_COST_CNT	MAINT_COST	AM0101_OCC
1	520	1
1	210	1
4	44	1
4	322	2
4	66	3
4	188	4
6	324	1
6	1103	2
6	566	3
6	755	4
6	988	5
6	1899	6
2	344	1
2	500	2

In some cases, the ORDER field may describe the months of the year (1 to 12). If all occurrences for the month of June are needed, for example, you would print data in which the ORDER field is equal to 6 (for June).

Using the preceding sample data, here is an example of how to use the ORDER field; this request always selects the sixth occurrence of maintenance costs:

```
TABLE FILE VEHICLES
PRINT MAINT_COST
WHERE AM0101_OCC EQ 6
END
```

```
MAINT_COST
-----
          1899
```

Using the same data, this request always selects the second occurrence of maintenance costs:

```
TABLE FILE VEHICLES
PRINT MAINT_COST
WHERE AM0101_OCC EQ 2
END
```

```
MAINT_COST
-----
          322
          1103
          500
```

Describing Multi-value Fields and Periodic Groups With the OCCURS Attribute

The OCCURS attribute is a segment attribute used to describe portions of records containing a multi-value field or a periodic group. When describing Adabas files, the OCCURS attribute must equal the two-character internal Adabas name appended with the letter C.

Records with repeating fields are defined to the server by describing the singly occurring fields in the parent segment, and the multiply occurring fields or groups in their own subordinate segment. The OCCURS attribute appears in the declaration for the subordinate segments.

The syntax is

```
OCCURS=occursname , $
```

where:

occursname

Indicates a data field in the parent segment that contains the number of occurrences of the descendant segment. Its value is derived from the two-character internal Adabas name (ALIAS field in the Access File) appended with the letter C.

The following is an example of the VEHICLES Master File, containing an MU segment with the counter field in the parent segment:

```

FILENAME=VEHICLES , SUFFIX=ADBSINX , $
SEGNAME=S01 , SEGTYPE=S , $
FIELD= REG_NUM , ALIAS= AA , A15 , A15 , INDEX=I , $
FIELD= CHASSIS_NUM , ALIAS= AB , I9 , I4 , $
FIELD= PERSONNEL_ID , ALIAS= AC , A8 , A8 , INDEX=I , $
FIELD= CLASS , ALIAS= AH , A1 , A1 , INDEX=I , $
FIELD= LEASE_PUR , ALIAS= AI , A1 , A1 , $
FIELD= CURR_CODE , ALIAS= AL , A3 , A3 , $
FIELD= MAINT_COST_CNT , ALIAS= AMC , I4 , I2 , $
GROUP= MODEL_YEAR_MAKE , ALIAS= AO , A28 , A22 , INDEX=I , $
FIELD= YEAR , ALIAS= AG , P2 , Z2 , $
FIELD= MAKE , ALIAS= AD , A20 , A20 , INDEX=I , $
SEGNAME=AM0101 , SEGTYPE=S , PARENT=S01 , OCCURS=AMC , $
FIELD= MAINT_COST , ALIAS= AM , P7 , P4 , $
FIELD= AM0101_OCC , ALIAS= ORDER , I4 , I2 , $

```

MAINT_COST is the MU field. It is allocated to the newly created segment, AM0101. The counter field, MAINT_COST_CNT, is located in the parent segment. The new segment, AM0101, uses this counter field to determine the number of occurrences of the MAINT_COST field (OCCURS=AMC).

Notice that the fields MAINT_COST_CNT and AM0101_OCC (ALIAS=ORDER) have USAGE=I4 and ACTUAL=I2. These are the required values for the USAGE and ACTUAL attributes of the counter field and the ORDER field.

The data adapter builds the PE segment in the same way. However, the component fields within the PE group are defined in the new segment.

The following is a view of the EMPLOYEES Master File, containing a PE group:

```

FILENAME=EMPFILE1 , SUFFIX=ADBSINX , $
SEGNAME=S01 , SEGTYPE=S , $
FIELD= PERSONNEL_ID , ALIAS= AA , A8 , A8 , INDEX=I , $
FIELD= FIRST_NAME , ALIAS= AC , A20 , A20 , $
FIELD= NAME , ALIAS= AE , A20 , A20 , INDEX=I , $
FIELD= LEAVE_BOOKED_CNT , ALIAS= AWC , I4 , I2 , $
SEGNAME=AW0401 , SEGTYPE=S , PARENT=S01 , OCCURS=AWC , $
GROUP= LEAVE_BOOKED , ALIAS= AW , A16 , A12 , $
FIELD= LEAVE_START , ALIAS= AX , P6 , Z6 , $
FIELD= LEAVE_END , ALIAS= AY , P6 , Z6 , $
FIELD= AW0401_OCC , ALIAS= ORDER , I4 , I2 , $

```

LEAVE_BOOKED is the PE group. It is allocated to the newly created segment, AW0401. The counter field, LEAVE_BOOKED_CNT, is located in the parent segment. The new segment, AW0401, uses this counter field to determine the number of occurrences of the LEAVE_BOOKED group (OCCURS=AWC).

The counter field, LEAVE_BOOKED_CNT, and the ORDER field, AW0401_OCC, require values of I4 for the USAGE attribute and I2 for the ACTUAL attribute.

Note: For both counter (CNT) and ORDER fields, Release 7.0.8 and higher supports ACTUAL=I2. Prior releases required ACTUAL=I1.

The number of occurrences is limited by the original definition of the file to Adabas.

To retrieve the appropriate occurrence of a repeating field in a report request, specify the ORDER field in your Master File. See *Specifying OCCURS Segment Sequence: The ORDER Field* on page 2-69 for details.

For detailed information about defining MU fields and PE groups in the Access File, see *Segment Attributes in Access Files* on page 2-47.

Describing Multi-value Fields Within Periodic Groups

This topic describes how to specify multi-value fields within periodic groups in the Master File. Define each multi-value field contained in the periodic group segment as a separate descendant segment of the periodic group.

A periodic group containing a multi-value field is defined as two separate segments.

All fields must be defined in the order in which they appear in the FDT.

Consider the EMPLOYEES file. It has a periodic group called INCOME, which contains one multi-value field, BONUS. Each counter field (for example, AQC) must be in its appropriate parent segment. The INCOME_CNT field is in the parent segment for the periodic group, INCOME. The BONUS_CNT field is in the parent segment for the MU field, BONUS. Any repeating fields within the periodic group are defined in the AQ0201 segment. Their corresponding Adabas two-character field names are the values for ALIAS with the appropriate values for USAGE and ACTUAL.

Here is the view of the Master File for this structure:

```

FILENAME=EMPFIL1 , SUFFIX=ADBSINX , $

SEGNAME=S01      , SEGTYPE=S , $
  FIELD= PERSONNEL_ID      , ALIAS= AA      , A8      , A8      , INDEX=I , $
  FIELD=FIRST_NAME        , ALIAS= AC      , A20     , A20     , $
  FIELD=NAME              , ALIAS= AE      , A20     , A20     , INDEX=I , $
1.  FIELD= INCOME_CNT      , ALIAS= AQC     , I4      , I2      , $

1.  SEGNAME=AQ0201      , SEGTYPE=S , PARENT=S01      , OCCURS=AQC , $
1.  $PEMU = INCOME      , ALIAS= AQ      , A19     , A13     , $
1.  FIELD=Curr_CODE      , ALIAS= AR      , A3      , A3      , $
1.  FIELD=SALARY        , ALIAS= AS      , P9      , P5      , $
2.  FIELD=BONUS_CNT      , ALIAS= ATC     , I4      , I2      , $
1.  FIELD= AQ0201_OCC    , ALIAS= ORDER   , I4      , I2      , $

2.  SEGNAME=AT0301      , SEGTYPE=S , PARENT=AQ0201  , OCCURS=ATC , $
2.  FIELD= BONUS        , ALIAS= AT      , P9      , P5      , $
2.  FIELD= AT0301_OCC   , ALIAS= ORDER   , I4      , I2      , $

```

1. PE fields.

This example includes the periodic group called INCOME, which contains the MU field, BONUS. CREATE SYNONYM comments out the group field with \$PEMU because its length is variable (depending on the number of occurrences of the MU field). The server cannot report on a group with an unknown length. However, the component fields can be referenced for reporting purposes.

2. MU fields.

BONUS_CNT (ALIAS=ATC) is the counter field. It is defined in the PE segment and tells us the number of occurrences for that field. Used in the MU segment, it indicates how many times this repeating field occurs, for example, OCCURS=ATC. Defined in the parent segment, the counter field is used by the child segment to tell us how many occurrences to expect.

Any periodic group segment must be described in its entirety. This description has two requirements: each field of the periodic group must have a field declaration in the periodic group segment, and each field in the group must be described in the order in which it appears in the Adabas FDT.

Specifying OCCURS Segment Sequence: The ORDER Field

There may be occasions with OCCURS segment processing when the order of the data is significant. For example, the field values may represent monthly or quarterly data, but the record itself may not explicitly specify the month (or quarter) to which the data applies. The ORDER field maintains the sequence number for each multiply occurring instance.

The order of the occurrences of a multi-value field or periodic group also has some significance in your application. For example, if a periodic group contains 12 occurrences, each occurrence could correspond to one month of the year. The first occurrence represents January, the second February, and so on through to December.

To use the ORDER option, you must include an additional field in your Master File with an ALIAS of ORDER. This option instructs the server to determine the sequence number of fields in an OCCURS segment. The server automatically supplies a value for the new field that defines the sequence number of each repeating group. The ORDER field does not represent an existing Adabas field; it is used only for internal processing.

The syntax rules for the ORDER field are:

- It must be the last field described in the OCCURS segment.
- The field name must be unique.
- The ALIAS value must be ORDER.
- The USAGE value must be I4, with any appropriate edit options.
- The ACTUAL value must be I2 (in earlier releases, I1 was used).

Each field with the ALIAS value ORDER tracks the order of occurrence of the preceding fields in the segment. Therefore, in the sample Master File in *Describing Multi-value Fields Within Periodic Groups* on page 2-67, AT0301_OCC maintains the sequence of BONUS occurrences.

An ORDER field may be decoded into a meaningful value in a DEFINE.

Using the GROUP Attribute to Cross-Reference Files

The GROUP attribute can be used to cross-reference Adabas files if the file you want to search does not contain a descriptor. This cross-referencing can be done *only* if fields within the file you want to search correspond to descriptors in the file you are cross-referencing.

Consider this situation: You have a SALES file which contains salesman information. It also has account information, such as COMPANY_CODE, ITEM_NUMBER, and ITEM_NAME.

Suppose you also have an ACCOUNT file containing information about each company in a salesman's territory. The ACCOUNT file has a superdescriptor consisting of COMP_CODE, IT_NUMBER, and IT_NAME. These fields correspond to the fields specified in the SALES file.

You can create a link between the matching fields in the SALES and ACCOUNT files. To do this, describe the matching fields from the SALES file as a group. Then you can join the group field to the superdescriptor in the ACCOUNTS file. The host file is the file, without a descriptor, containing the held values which must be grouped in order to retrieve related records in the second file. For example:

```
FILE=SALES      ,SUFFIX=ADBSINX      ,$
  SEGNAME=ACCT_SEG  ,SEGTYPE=S      ,$
  GROUP=LINKFLDS      ,ALIAS=      ,USAGE=A20 ,ACTUAL=A20 ,$
    FIELD=COMPANY_CODE  ,ALIAS=BA ,USAGE=A3  ,ACTUAL=A3  ,$
    FIELD=ITEM_NUMBER  ,ALIAS=BB ,USAGE=A5  ,ACTUAL=A5  ,$
    FIELD=ITEM_NAME    ,ALIAS=BC ,USAGE=A12 ,ACTUAL=A12 ,,$
```

SALES is the host file which uses the superdescriptor (COMPANY) in the ACCOUNT file for the company-related data. The superdescriptor in the ACCOUNT file, easily recognized by TYPE=SPR in the Access File, is composed of fields that correspond to the dummy group in the host. Looking at the GROUP attribute used to cross-reference files, above and the GROUP attribute used to cross-reference files, below, you see how the fields in the LINKFLDS group in the SALES file relate to the superdescriptor COMPANY in the ACCOUNT file:

```
FILE=ACCOUNT      ,SUFFIX=ADBSINX      ,$
  SEGNAME=PROD_SEG  ,SEGTYPE=S ,PARENT=ACCTS  ,$
  GROUP=COMPANY      ,ALIAS=S1 ,USAGE=A20 ,ACTUAL=A20 , INDEX=I ,,$
    FIELD=COMP_CODE  ,ALIAS=AA ,USAGE=A3  ,ACTUAL=A3  ,$
    FIELD=IT_NUMBER  ,ALIAS=AB ,USAGE=A5  ,ACTUAL=A5  ,$
    FIELD=IT_NAME    ,ALIAS=AC ,USAGE=A12 ,ACTUAL=A12 ,,$
```

Use the GROUP superdescriptor COMPANY to retrieve data for those values using the JOIN command within a procedure. The JOIN command or embedded cross-reference completes the link. In this example, you would join LINKFLDS in SALES to COMPANY in ACCOUNT using the following syntax:

```
JOIN LINKFLDS IN SALES TO ALL COMPANY IN ACCOUNT AS J1
```

Platform-Specific Functionality

The Data Adapter for Adabas on UNIX, Windows, and OpenVMS, has more restrictions than on OS/390 and z/OS. These restrictions are described below.

Note: On all platforms, if an error occurs while Adabas commands are being processed, then the communication block (CB) is returned along with the Response Code (RC) Additions 2 field, in 2 or 4 bytes binary format. The Additions 2 field indicates the specific reason for the Response Code. The Adabas Adapter displays both RC and Additions 2 fields in the error message. This information is available in the trace as well.

Reference **Functionality on UNIX and Windows**

The following conditions apply when using the adapter on UNIX and Windows platforms:

- You cannot use non-descriptor fields as search criteria.
- The parameter NDFIND is automatically set to the OFF value if the adapter is executed in a non-mainframe environment. If non-descriptor fields are used as key fields, the entire database is read.
- If the Master File non-descriptor field is declared as an indexed field (INDEX=I or FIELDTYPE=I) and used as a key field, the result from Adabas is RC 61.
- If an occurrence of an MU (multi-value) field is deleted, the adapter shifts all other occurrences into its place and assigns an empty value for the last occurrence. If this field has an NU (null suppression) option, the last occurrence will be automatically deleted by Adabas.
- DBNO must be assigned to a valid numeric value for the existing database.

Reference **Functionality for OpenVMS Platform**

Adabas on OpenVMS does not allow a direct request for sub/superdescriptors values. For this reason, there are some limitations on using these fields (with TYPE=NOP/SPR in ACX) in server requests to Adabas:

- NOP fields cannot be printed or displayed, even if they are alphanumeric.
- NOP/SPR fields cannot be used in functions (MIN, MAX, and so on).
- If NOP/SPR fields are used in the IF/WHERE criteria, then CALLTYPE=RL is automatically changed to CALLTYPE=FIND. It can change the sequence of returning data rows on OpenVMS in comparison with sequence on other platforms.
- NOP fields cannot be used in a complex IF/WHERE clause.
- NOP fields cannot be used for joining.

Customizing the Adabas Environment

The Data Adapter for Adabas provides several parameters for customizing the environment and optimizing performance.

The sections *Multifetch and Prefetch* on page 2-74, *Adabas Dynamic Database Number* on page 2-76, and *Running in 24-Bit Mode* on page 2-77, describe the environment commands that display and change the parameters governing your server session.

ORDER Fields in the Indexed Field List

ORDER fields maintain the sequence number for each multiply occurring instance. The default setting is ON.

Setting ORDERKEYS to OFF forces the Data Adapter for Adabas to exclude ORDER fields from the list of indexed fields.

Syntax How to Set ORDERKEYS in the Indexed Field List

```
ENGINE ADBSINX SET ORDERKEYS {ON|OFF}
```

Switching the Access Mode Using NOPACCESS

This setting allows you to switch the access mode from RL to FIND if field defined with TYPE=NOP was selected (even if CALLTYPE=RL has been specified).

The default value for this setting is MIXED. A new setting can be useful when the NOP super descriptor defined in a root segment is derived from fields that belong to different Master File segments.

Syntax How to Set NOPACCESS

```
ENGINE ADBSINX SET NOPACCESS FIND
```

NEW Synonym Setting for Superdescriptors

Setting SYNONYM to NEW instructs the Data Adapter for Adabas to use new logic when:

- Describing metadata
- Processing requests that can take advantage of superdescriptors for screening conditions.

Note: This command must be specified in the EDASPROF.PRF file.

Syntax **How to Set Synonyms for Superdescriptors**

```
ENGINE ADBSINX SET SYNONYM {NEW|STD}
```

NEW

Describes superdescriptors as groups in the Master File. With NEW synonym format, neither the groups nor their components (fields) have field names specified in the Master File. Aliases are populated using the Adabas FDT (2-byte name).

When a synonym contains the NEW syntax, the Data Adapter for Adabas uses NEW logic to process the selection criteria of requests involving that synonym. Screening conditions within the request are analyzed and the superdescriptor that covers the highest order component fields required by the selection criteria are used.

Note that you must specify CALLTYPE =RL in the Access File in order to take advantage of superdescriptor-based access.

STD

Incorporates the usual CREATE SYNONYM behavior, which does not trigger the new logic for requests using superdescriptors. STD is the default value.

For more information on synonyms, see *Creating Synonyms* on page 2-22.

Example **Setting Synonyms for Superdescriptors**

The following Master File illustrates superdescriptors Y6 and Y7 defined as GROUP, with the NEW synonym format:

```
GROUP =                ,ALIAS=Y6                ,A2      ,A2      ,INDEX=I , $
  FIELD=                ,ALIAS=AF                ,A1      ,A1      ,         , $
  FIELD=                ,ALIAS=AG                ,A1      ,A1      ,         , $
GROUP =                ,ALIAS=Y7                ,A22     ,A22     ,INDEX=I , $
  FIELD=                ,ALIAS=AF                ,A1      ,A1      ,         , $
  FIELD=                ,ALIAS=AG                ,A1      ,A1      ,         , $
  FIELD=                ,ALIAS=AC                ,A20     ,A20     ,         , $
```

The corresponding Access File specifies the superdescriptors Y6 and Y7 and their components, using the keywords SUPER and NUMFLDS:

```
SUPER= Y6                ,NUMFLDS=2                , $
  FIELD=AF_FIELD        ,TYPE=          ,NU=NO        , $
  FIELD=AG_FIELD        ,TYPE=          ,NU=NO        , $

SUPER= Y7                ,NUMFLDS=3                , $
  FIELD=AF_FIELD        ,TYPE=          ,NU=NO        , $
  FIELD=AG_FIELD        ,TYPE=          ,NU=NO        , $
  FIELD=AC_FIELD        ,TYPE=          ,NU=NO        , $
```

SUPER specifies the native Adabas two byte field name of the superdescriptor.

NUMFLDS defines the number of participating fields. Descriptions of all participating fields immediately follow the SUPER statement.

The request that follows uses superdescriptor Y6 in the L3 (Read Logical) command in the Access File:

```
TABLE FILE EMP211
PRINT AF_FIELD AG_FIELD AC_FIELD AD_FIELD
IF AF_FIELD EQ 'S'
IF AG_FIELD EQ 'M'
END
```

The next request uses superdescriptor Y7 in the L3 (Read Logical) command. Notice that it references an extra field in the last line of the Access File:

```
TABLE FILE EMP211
PRINT AF_FIELD AG_FIELD AC_FIELD AD_FIELD
IF AF_FIELD EQ 'S'
IF AG_FIELD EQ 'M'
IF AC_FIELD EQ 'C'
END
```

Multifetch and Prefetch

The Adabas Multifetch and Prefetch options reduce execution time and allow Adabas data to be retrieved with a high degree of efficiency. By buffering multiple record results from a single call, Multifetch and Prefetch reduce the communication overhead between the Data Adapter for Adabas and the Adabas nucleus.

Adabas Release 5.3.2 is the first release to support Multifetch, and Release 4.0 is the first release to support Prefetch. The adapter uses the Multifetch option if it is available (this is the default option), or the Prefetch option if it is available. The option available is determined by your site's environment.

The data adapter trace file FSTRACE4 contains the following information about the Fetch option:

- Fetch is allowed or disallowed.
- Fetch technique being used (Multifetch or Prefetch).
- Number of records per Fetch buffer.
- Size of the Fetch buffer (ISN buffer).

The Multifetch and Prefetch options require that OPEN=YES (the default value) is specified in the Access File. The Multifetch and Prefetch options are activated as long as OPEN=YES is specified. There are two ways to deactivate (or reactivate) these options:

- Issue SET commands before running the request.
- Include the FETCH and FETCHSIZE attributes in the Access File.

Note: The SET commands override the Access File settings. This setting will be in effect for the entire server session.

Syntax **How to Set FETCH and FETCHSIZE**

The syntax for the SET commands is

```
ENGINE ADBSINX SET FETCH {ON|OFF|DEFAULT}
ENGINE ADBSINX SET FETCHSIZE {n|MAX[IMUM]}
```

where:

ENGINE

Indicates the OS/390 and z/OS operating system.

ON

Sets the Fetch feature on for the user session.

OFF

Sets the Fetch feature off for the user session.

DEFAULT

Uses the information from the Access File. This value is the default.

n

Is the number of records per buffer (1–999). The buffer size varies with the size of the record and can be different for each TABLE request. The buffer size limit is 32K. The default is 10 records per buffer.

MAX[IMUM]

Is automatically calculated by the Data Adapter for Adabas to allow the maximum number of records that fit in a 32K buffer.

Syntax **How to Declare the FETCH and FETCHSIZE Attributes in the Access File**

FETCH and FETCHSIZE are applicable only to segments described as ACCESS=ADBS in the Access File.

The Access File can contain the following attributes in the SEGNAM statement

```
FETCH = {ON|OFF}
FETCHSIZE = {n|MAX[IMUM]}
```

where:

ON

Sets the Fetch feature on for the user session. This value is the default.

OFF

Sets the Fetch feature off for the user session.

n

Is the number of records per buffer (1 to 999). The default is 10 records per buffer.

MAX[IMUM]

Is automatically calculated by the Data Adapter for Adabas to allow the maximum number of records that fit in a 32K buffer.

Here is an example of the FETCH and FETCHSIZE attributes in the sample Access File for EMPLOYEES:

```
$$$ FILENAME=EMPFILE1,SUFFIX=ADBSINX,$  
RELEASE=6,OPEN=YES,$
```

```
$ ADABAS FILE = EMPLOYEES                                DICTIONARY =  
SEGNAM=S01 ,ACCESS=ADBS,FILENO=001,CALLTYPE=RL,  
                SEQFIELD=PERSONNEL_ID,FETCH=ON,FETCHSIZE=15,$  
FIELD= DEPT_PERSON          ,TYPE=SPR,$  
FIELD=DEPT_S03              ,TYPE=DSC,NU=YES,$  
FIELD=NAME_S03              ,TYPE=    ,NU=NO,$
```

See your Software AG documentation for more information about the Multifetch/Prefetch feature.

Adabas Dynamic Database Number

Previous versions of the Data Adapter for Adabas required specification of the Adabas database number in the Access File using the DBNO attribute. This feature required a duplicate copy of the Access File for each database accessed.

Adabas database numbers can now be set from the server's profile. A new SET command allows users to override the DBNO in the Access File. Note that specifying database numbers in the Access File is still supported. Use of the dynamic database number makes Master and Access Files shareable among databases.

The SET command remains valid throughout the server session.

Syntax

How to Set the Adabas Dynamic Database Number

```
ENGINE ADBSINX SET DBNO dbno
```

```
ENGINE ADBSINX SET DBNO dbno AFD afd
```

```
ENGINE ADBSINX SET DBNO dbno AFD afd SEG[NAM] segname
```

```
ENGINE ADBSINX SET ?
```

```
ENGINE ADBSINX SET DBNO DEFAULT
```

where:

```
ADBSINX
```

Indicates the Data Adapter for Adabas.

dbno

Is any valid numeric database value between 0 and 255.

afd

Is any valid Access file name.

segname

Is any valid ADBS segname in the Access File.

?

Queries the current settings.

DEFAULT

Returns to the default settings for all previous settings. The DBNO is read from the Access File. If the attributes are not specified in the Access File, the data adapter will determine the DBNO from the DDCARD.

Examples for OS/390 and z/OS are shown below:

<code>ENGINE ADBSINX SET DBNO 5</code>	Retrieves all data from database number 5.
<code>ENGINE ADBSINX SET DBNO 2 AFD EMPFILE</code>	Retrieves data for the EMPFILE from database number 2.

Note: Several commands can be issued for different files. Each command must be issued separately. Changes are cumulative. For example:

```
ENGINE ADBSINX SET DBNO 1 AFD EMPFILE
ENGINE ADBSINX SET DBNO 2 AFD EMP2
ENGINE ADBSINX SET DBNO 3 AFD VEHICLES
```

<code>ENGINE ADBSINX SET DBNO 2 AFD EMPFILE SEG S02</code>	Retrieves data for the EMPFILE in database number 2 for a particular ADBS segment (segment S02).
<code>ENGINE ADBSINX SET ?</code>	Displays your settings.
<code>ENGINE ADBSINX SET DBNO DEFAULT</code>	Returns to the default settings. This resets all SET DBNO commands for all Access Files.

Running in 24-Bit Mode

The following command is used to allocate buffers below the line because of certain user exits that must run below the line. Issue this command in your server session:

```
ENGINE ADBSINX SET AMODE 24
```

Adabas Reporting Considerations

The server report procedure designers have great flexibility in coding the description of the part of the Adabas structure they want to access. These topics describe the methods of file traversal that the Data Adapter for Adabas uses to determine the relative advantages of different file descriptions.

Adabas File Navigation Techniques

When you define specific hierarchical representations of Adabas structures in Master Files, you specify the order in which you want the Data Adapter for Adabas to retrieve records. This procedure is called navigational logic and is usually part of the application program. Navigation techniques using the JOIN command also work this way.

Referencing Subtrees and Record Retrieval

The Data Adapter for Adabas selects where to enter the subtree, called the point-of-entry, and the subsequent navigational processing by analyzing the tree structure defined by your Master File (or JOIN structure) and report request. The data adapter determines the smallest subtree that contains all the fields needed for retrieval to produce a report.

The smallest subtree is composed of those segments that contain fields referenced by the request, plus the minimum number of additional segments required to connect all the files used in the request.

The adapter retrieves records only in segments in the referenced subtree. Within the subtree, it retrieves records that contain fields required for the report request or records that are needed to provide the correct links between report fields.

The adapter always enters a database through the root segment of the referenced subtree.

Segment Retrieval Methodology

The Data Adapter for Adabas retrieves segments from top-to-bottom, then left-to-right at each level of the hierarchy. It retrieves all unique descendant segments before any non-unique descendant segments.

This treatment of unique segments is consistent with a basic server principle: for reporting purposes (though not for updating or file organization), a unique child is logically a direct extension of its parent. This principle is an important factor in selecting a structure that properly reflects your Adabas file. The results of SUM and COUNT operations on fields in child segments may depend on whether they have been declared unique or non-unique. The server also treats missing segments differently, depending on whether the segment is declared unique or non-unique.

Missing Unique Segments

If a segment is specified as unique (SEGTYPE=U or KU), the server regards it as a logical extension of the parent segment. The Data Adapter for Adabas automatically inserts default values (blanks for alphanumeric fields and zeros for numeric fields) if the unique child segment does not exist. As a result, unique segments are always present.

Syntax **How to Use Missing Non-Unique Segments**

If a segment is specified as non-unique (SEGTYPE=S or KM), select one of three options for processing a record without descendant segments.

The syntax is

```
SET ALL = all_option
```

where:

all_option

Allows for the processing of records with no descendant segments. Possible values are:

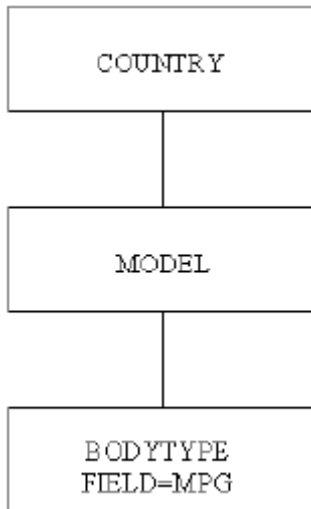
OFF which omits parent instances that are missing descendant segments from the report. This value is the default.

ON which includes parent instances that are missing descendant segments in the report.

PASS which includes parent instances that are missing descendant segments, even when IF statements exist to screen fields in the descendant segment's missing instances.

SET ALL may be specified in a profile or procedure.

The examples in the following topics describing the SET ALL command are based on the following structure:



SET ALL=OFF

The default option (SET ALL=OFF) rejects a record if the request calls for retrieval of a descendant segment and there is no descendent segment present.

For example, assume you have a file in which the parent segment is COUNTRY, which has a descendant segment named MODEL, which in turn has a descendant segment named BODYTYPE. Using the SET ALL=OFF option, the statement

```
COUNT BODYTYPE BY COUNTRY
```

does not print in the report the details of any country that did not produce at least one bodytype of a model of a car.

SET ALL=ON

The Data Adapter for Adabas displays the parent record, even if it has no descendant segments. In this case, using the SET ALL=ON option when processing the statement

```
COUNT BODYTYPE BY COUNTRY
```

displays the names of all countries and gives a count of zero (0) bodytypes for those without descendant segments.

However, if the request has an explicit screening test on the descendant segment, the absence of any descendant segments results in test failure. For example, the request

```
COUNT BODYTYPE BY COUNTRY  
IF MPG GT 22
```

excludes any country without any bodytype segments from the report.

SET ALL=PASS

The third option, SET ALL=PASS, allows parents without a descendant segment to pass an explicit screening test on that descendant segment. The request

```
COUNT BODYTYPE BY COUNTRY
```

lists all countries with or without bodytype segments. The request

```
COUNT BODYTYPE BY COUNTRY  
IF MPG GT 22
```

includes records without any bodytype segments, and those with an MPG greater than or equal to 22.

The ALL Prefix

Selectively apply SET ALL=ON by adding the ALL prefix to any field from the desired segment.

Reference the field either as a sort field (for example, BY ALL.COUNTRY or ACROSS ALL.COUNTRY) or as a verb object (WRITE ALL.COUNTRY). As a result, the SET ALL=ON strategy is applied to any missing, immediate, non-unique descendants of the segment containing the ALL-prefixed field. The SET ALL=OFF option is in effect for all other segments.

For example, in the request

```
COUNT MODEL AND BODYTYPE BY ALL.COUNTRY
```

the SET ALL=ON option applies to the country segment and its descendant segments. Therefore, if there is a country without models (and consequently without bodytypes), the report shows that country. Any test condition on either the model or the bodytype segment nullifies the effect of the ALL prefix.

The global SET ALL settings of ON and PASS take precedence over the selective ALL prefix. The selective ALL prefix is effective only when the global setting is OFF, either explicitly or by default.

Adabas Selection Considerations

The Data Adapter for Adabas analyzes all selection criteria that apply to a specific report request and uses the criteria to minimize its search for data. If a record fails any of the selection tests, the server does not attempt to retrieve any descendant records. Retrieval continues with the next record in the parent segment. If there is no other record in that parent segment and it is not the root of the Master File, the server moves back up to the next record in the previous segment.

The selection tests that you impose on a high-level segment are much more efficient at reducing I/O operations than criteria imposed on lower level segments. If you know in advance which selection criteria are likely to be used most frequently, design the Master File to take advantage of the hierarchical structure in the Data Adapter for Adabas.

Selection Order in the Access File

When a report request contains multiple optimizable selection tests, the order of descriptors in the Access File determines the order in which the server applies the selection tests. The server issues a Read Logical (RL) call using the first descriptor listed in the Access File that participates in a selection test.

Therefore, for efficient processing you should describe the most restrictive descriptor at the beginning of its segment in the Access File. The order of descriptors in the Master File has no effect on selection processing.

Syntax **How to Use RECORDLIMIT and READLIMIT**

For any request, you may limit the number of Adabas records retrieved from the database and the number of read operations performed. Use the RECORDLIMIT and READLIMIT keywords to impose these limitations by adding the following conditions to a report request

```
{WHERE|IF} RECORDLIMIT {EQ|IS} n
{WHERE|IF} READLIMIT {EQ|IS} n
```

where:

n

Is the number of records or read operations at which you want to limit the search.

You add these conditions to any report request, or incorporate them into file security through the DBA facility.

Consider the following example

```
TABLE FILE VEHICLES
PRINT PERSONNEL_ID MAKE MODEL YEAR
WHERE RECORDLIMIT EQ 5
END
```

which produces this report:

```
PAGE          1

PERSONNEL_ID  MAKE          MODEL          YEAR
-----      -
11500330     CITROEN          BX LEADER      85
11400313     ALFA ROMEO      SPRINT GRAND  84
30034217     AUSTIN          MINI           80
30034228     TALBOT          SOLARA        83
30008427     AUSTIN          MINI           80
```

Notice that only five records are reported as requested.

Syntax

How to Use Optimization of the FIND Call Using Non-Descriptor Fields

It is possible to use non-descriptor fields as search criteria and have the calls to Adabas use the search buffer rather than read through the entire database.

The Data Adapter for Adabas provides improved optimization by allowing the search buffer to be generated using non-descriptor fields. This optimization occurs whenever CALLTYPE= FIND is specified in the Access File and you include an IF or WHERE test referencing a non-descriptor field in your report request.

It may prove to be more efficient to alter your retrieval strategy and perform a Read Physical (L2) call when large amounts of data exist. A SET command is provided for changing the default Adabas call when selecting non-descriptor fields.

The syntax is

```
ENGINE ADBSINX SET NDFIND {ON|OFF}
```

where:

ENGINE

Indicates the OS/390 and z/OS operating system.

ON

Causes the search buffer to be generated with any field (non-descriptor and/or descriptor field). This value is the default.

OFF

Causes the search buffer to be generated with only descriptor fields. If the request does not use any descriptor field, the Read Physical call is generated.

This command applies only if CALLTYPE= FIND is specified in the Access File.

Using the JOIN Command to Process Multiple Files

You can JOIN Adabas databases defined to the server to other Adabas databases or to any other fully joinable database that the server can read. Each JOIN creates a parent-child relationship. The parent field is called the host or *from* field. The child field is called the cross-referenced or *to* field.

You can JOIN to Adabas databases if the cross-referenced field is one of the following:

- A descriptor field.
- A superdescriptor defined with TYPE=SPR or NOP in the Access File.
- A subdescriptor defined with TYPE=NOP in the Access File.

In every case, in the Access File, the cross-referenced segment must specify ACCESS=ADBS.

If CALLTYPE=RL is specified for the cross-referenced segment, the host field can be joined to the high-order portion of a descriptor, superdescriptor, or subdescriptor.

When an Adabas file is the host file, the host field is one of the following:

- A non-recurring field (ACCESS=ADBS).
- A field in a periodic group (ACCESS=PE).
- A multi-value field (ACCESS=MU).

The Data Adapter for Adabas also supports DEFINE-based JOINS. Up to 16 JOINS can be in effect at one time.

Example Multi-field JOIN and Short-to-Long JOIN Capability

For a multi-field JOIN, the number of fields used in the JOIN must be the same for both the host and the cross-referenced files. The cross-referenced fields must describe the left-most portion of a superdescriptor defined to the server using the GROUP attribute. Consider the following example.

```
JOIN FLDA AND FLDB IN ADBS1 TO KEY1 AND KEY2 IN ADBS2 AS J1
```

For the short-to-long JOIN, the cross-referenced field must be a descriptor, subdescriptor, or superdescriptor, or a field that describes the left-most portion of a GROUP superdescriptor.

Adabas Optimization With Null-Suppression for CALLTYPE=RL

In the Data Adapter for Adabas, optimization refers to using an index to retrieve the answer set. To ensure data integrity and complete answer sets, the Data Adapter for Adabas will perform optimization when:

- All non-referenced component fields of a superdescriptor are *not* null-suppressed (NU=NO in the Access File).
- All component fields of a superdescriptor that contain null-suppressed fields (NU=YES in the Access File) are used in the selection criteria.

If a field is null-suppressed in Adabas (NU=YES in the Access File), any superdescriptor that uses this field has no entry on the inverted list when this field is blank (alphanumeric) or zero (numeric).

Master File With a Three-Field Superdescriptor

```
GROUP=SUPERG ,ALIAS=S1 ,USAGE=A9 ,ACTUAL=A9 ,INDEX=I ,  
FIELD=FLD1 ,ALIAS=AA ,USAGE=A3 ,ACTUAL=A3 ,  
FIELD=FLD2 ,ALIAS=AB ,USAGE=A3 ,ACTUAL=A3 ,  
FIELD=FLD3 ,ALIAS=AC ,USAGE=A3 ,ACTUAL=A3 ,
```

Access File With a Three-Field Superdescriptor

```
FIELD= SUPERG ,TYPE=SPR ,  
FIELD= FLD1 ,TYPE= ,NU=YES ,  
FIELD= FLD2 ,TYPE= ,NU=NO ,  
FIELD= FLD3 ,TYPE= ,NU=YES ,
```

In order to optimize a selection test against a superdescriptor with null-suppression, you *must explicitly reference the null-suppressed field* in the superdescriptor. If you do not reference the null-suppressed field and the field has no data, there is no record in the index and optimization would return no records. To ensure correct results, the server will not optimize the selection test if the null-suppressed field is not referenced.

For more information about null-suppression and how it affects data retrieval, see your Software AG documentation.

Adabas Optimization on Group Fields

If a report request contains IF or WHERE selection tests against one or more fields that describe the left-most portion of a GROUP descriptor, the Data Adapter for Adabas combines this request into a test that uses the superdescriptor for greater efficiency. If any of the component (or parent) fields of the superdescriptor are defined to Adabas with null-suppression, be sure to note this information in the Access File to ensure accuracy of reads.

For example, consider the following Master File extract:

```
GROUP=UPERD ,ALIAS=SD ,USAGE=A8 ,ACTUAL=A8 ,INDEX=I , $
  FIELD=PART1 ,ALIAS=P1 ,USAGE=A4 ,ACTUAL=A4 , $
  FIELD=PART2 ,ALIAS=P2 ,USAGE=A4 ,ACTUAL=A4 , $
```

If, in a report request, you include the following two tests,

```
WHERE PART1 EQ 'ABCD'
WHERE PART2 EQ 'EFGH'
```

these two tests are equivalent to the following syntax:

```
WHERE SUPERD EQ 'ABCD/EFGH'
```

This combination uses the superdescriptor's inverted list and optimizes the Adabas call. This optimization is performed only if all null-suppressed (NU=YES) superdescriptor components are explicitly referenced in IF or WHERE tests.

Test on Group Field With Numerics

If you are testing on a group that contains numeric fields, the test must contain the sign byte. The Data Adapter for Adabas passes only one value per numeric field, based on the preferred sign values in Adabas. A sign value of F is passed for positive numbers and a sign value of D is passed for negative numbers. This sign value reduces the number of calls sent to Adabas and also eliminates the need for Adabas to perform any logical sign translation.

For example:

```
GROUP=GROUP1 ,ALIAS=S1 ,USAGE=A9 ,ACTUAL=A14 ,INDEX=I , $
  FIELD=FLD1 ,ALIAS=AA ,USAGE=A3 ,ACTUAL=A3 , $
  FIELD=FLD2 ,ALIAS=AB ,USAGE=P3 ,ACTUAL=P3 , $
  FIELD=FLD3 ,ALIAS=AC ,USAGE=A3 ,ACTUAL=A3 , $
```

In a report request, you must include the sign for the P3 field:

Mainframe platforms:

```
WHERE GROUP1 EQ 'ABC/123F/XYZ'
```

Non-mainframe platforms:

```
WHERE GROUP1 EQ 'ABC/123/XYZ'
```

Adabas Writing Considerations

The Data Adapter for Adabas is designed to support SQL update commands for an Adabas data source without the use of remote procedures. These topics describe the methods and rules related to write capability.

Adabas Write Data Adapter

Before you can use any commands that write to an Adabas data source, you must make the following changes to the Master and Access Files:

In the Master File:

- All segments must have SEGTYPE = S0 (for the Read Data Adapter, SEGTYPE=S is sufficient).
- Fields with ACTUAL format Z (zoned) must have a numeric USAGE format (P or I). Format A is supported only for reading an Adabas data source.

Note: For the best performance, you can change ACTUAL format Z to ACTUAL format P in the Master File. In this case, you must also change the ALIAS attribute to contain the length and type. If the ALIAS was ff, and the field length is III, the ALIAS attribute should be coded as:

```
ALIAS = 'ff,III,P'
```

For example, consider the following field declaration:

```
FIELD = LEAVE_DUE ,ALIAS = AU ,USAGE = P2 ,ACTUAL = P2 , $
```

You would edit this declarations as follows:

```
FIELD = LEAVE_DUE ,ALIAS = 'AU,2,P' ,USAGE = P2 ,ACTUAL = P2 , $
```

- If two or more fields in the Master File are synonyms (they refer to the same field in the data source and, therefore, have the same ALIAS attribute), only the first field encountered in the Master File can be used in INSERT and UPDATE commands. If a synonym other than the first is used in an INSERT command, it is ignored. If one is used in an UPDATE command, processing is terminated with the following error message:

```
(FOC4565) IGNORED ATTEMPT TO CHANGE NONUPDATABLE FIELD
```

In the Access File:

- The segment attribute WRITE=YES indicates that values of fields from the segment may be modified.
- If a segment has the attribute ACCESS=ADBS, you can define a unique key by adding the following attribute:

```
UNQKEYNAME=name
```


where:

name

Is the name of the elementary or group field to be used as the unique key.

The UNQKEYNAME attribute does not necessarily define the key described in the ADABAS FDT (the UQ option). The data adapter uses it to decide which rules to apply in an INSERT, DELETE, or UPDATE command. If the UNQKEYNAME attribute does not correspond to the key described with the UQ option in the ADABAS FDT, Adabas and the data adapter may not agree on whether a segment instance is unique. This can affect the results of INSERT commands. If this attribute is not present, the data adapter uses the rules for modifying a segment with a non-unique key or no key. If it is present, the data adapter uses the rules for modifying a segment with a unique key. Subsequent sections describe these rules.

- We recommend the use of CALLTYPE=FINN instead of CALLTYPE=RL.

When using the Write Data Adapter, the data adapter automatically sets the values of the following two options:

- ADABAS OPEN is set to YES.
- FETCH is set to OFF for all segments.

SQL Commands for Data Adapter for Adabas

The Data Adapter for Adabas supports the following commands:

- SQL INSERT
- SQL UPDATE

Note: Certain types of fields listed in Master File cannot be updated. For more information, see *Fields That Cannot be Updated* on page 2-89.

- SQL DELETE

The adapter issues a COMMIT after each INSERT, UPDATE, or DELETE call. If the Adabas process fails, the adapter issues a ROLLBACK. User rollback of the Adabas process is not supported for SQL commands.

Reference Fields That Cannot be Updated

The SQL UPDATE command for the following types of fields is ignored:

- Counter fields (ALIAS=xxC).
- ORDER fields (ALIAS=ORDER).
- Group fields.
- Unique key fields (fields specified by the UNQKEYNAME attributes in the Access File).

The SQL UPDATE command for the following types of fields produces an error:

- Synonym fields.
- Fields created from sub- or superdescriptors (as defined in the Access File).

Reference Using Synonym Fields (Not Related to CREATE SYNONYM)

In a Master File, two or more field declarations can refer to the same Adabas field. Each duplicate field declaration after the first is called a synonym field. Such synonym fields can be used in report commands, but cannot be used in write commands. The following actions occur as a result of using synonyms in write commands:

- Synonym fields used in SQL INSERT are ignored.
- Synonym fields used in SQL UPDATE cause processing to terminate and generate the error message:

(FOC4565) IGNORED ATTEMPT TO CHANGE NONUPDATABLE FIELD

Syntax How to Insert Records Into an Adabas Data Source

```
INSERT INTO mfname [(field1, field2, ...)]  
VALUES ('value1', 'value2', ...)
```

where:

mfname

Is the name of the Master File to use.

field1, *field2*, ...

Is an optional list of fields to be inserted. If you omit the list, the value list must contain values for all fields in all segments in the Master File as long as the Master File only specifies a single path Adabas file. If the Master File is multi-path, the field list is required and can specify only a single path. Elementary fields not included in the field list have empty values in the data source. The field list can contain only elementary fields. Groups, sub-descriptors, super-descriptors, and hyper-descriptors, cannot be used in the field list.

'*value1*', '*value2*', ...

Is the list of values to be inserted. Each alphanumeric value must be enclosed in single quotation marks. If you specify the field name list, the value list only needs to specify, in field name list order, the values for the field names supplied. At a minimum, you must supply all the key fields for all the segments in the path to the target.

If a segment is not present, default values are generated for all fields that are not supplied, and the missing path segments are inserted along with the target segment.

Reference **Rules for Inserting Records Into an Adabas Data Source**

- For a segment with a unique key:

The key field value is used to insert the target segment. If any additional fieldname=value pairs are supplied for a segment in the path, they are used to qualify that path segment. If a segment with ACCESS=ADBS has a unique key or group of keys, only these key fields should be used in the INSERT command. All other fields should be added to the record using the UPDATE command.
- For a segment with a non-unique key or no key:

If you want to insert an additional record for an existing key field, you must have at least one field in addition to the key field in the field list in order to make the record unique. If you do not, the insert is rejected.
- For segments with ACCESS=PE or MU, if a new occurrence is inserted, you must set the occurrence number (ORDER field) to 0 (zero indicates the next occurrence) or to a value greater than the number of existing occurrences. This new occurrence is always inserted after the last existing occurrence. For example, if a PE or MU segment has two existing occurrences, the next occurrence added is always the third. If the occurrence with the given number already exists, processing terminates with the following message:

`(FOC4564) THIS OCCURRENCE ALREADY EXISTS. USE UPDATE COMMAND`

You should use the UPDATE command instead of INSERT in this case.
- For segments in which the parent segment contains ACCESS=PE, and the child segment contains ACCESS=MU without the NU option in the ADABAS FDT:
 - If an INSERT command is issued for the parent (PE) segment alone, Adabas automatically inserts an empty child (MU) segment. You can use the UPDATE command to provide values for this child.
 - If one INSERT command is issued for the parent (PE) and child (MU) segments simultaneously, the first occurrence in the child segment is inserted by the Write Data Adapter using the values from the INSERT command.
- For segments in which the parent segment has ACCESS=PE, and the child segment has ACCESS=MU with the NU option in the ADABAS FDT, the empty child is automatically suppressed by Adabas. You can use an INSERT command for the parent and child segments separately or simultaneously.

Reference **The Effect of UNQKEYNAME on INSERT Actions**

The UNQKEYNAME attribute in the Access File determines how the data adapter presents an INSERT command to Adabas. The UQ option in the ADABAS FDT and the specific field values listed in the INSERT command determine whether Adabas actually inserts the segment instance. The following table describes how these factors affect the result of the INSERT command. Assume that the Access File specifies UNQKEYNAME=EMPLOYEE_ID and that the employee ID value EMPID005 already exists in the data source:

Result of the INSERT Command for Existing EMPLOYEE_ID EDMPID005

EMPLOYEE_ID has the UQ Option in FDT	Fields in INSERT Command	Instance Inserted
No	EMPLOYEE_ID only.	No - rejected duplicate
Yes	EMPLOYEE_ID only.	No - rejected duplicate
No	EMPLOYEE_ID plus fields with values that do not already exist.	Yes
Yes	EMPLOYEE_ID plus fields with values that do not already exist.	No - error (FOC4561), RC=198

Result of INSERT Command when EMPLOYEE_ID is not in the field list

UNQKEYNAME = EMPLOYEE_ID	Instance Inserted
Yes	No - error (FOC4563)
No	Yes, with empty EMPLOYEE_ID value.

Syntax **How to Update Field Values in an Adabas Data Source**

```
UPDATE mfname SET field1 = 'value1' [,field2 = 'value2'...]
WHERE kfield1 = 'kvalue1' [AND kfield2 = 'kvalue2'...]
```

where:

mfname

Is the name of the Master File to use.

field1, field2, ...

Are the names of the fields to be updated.

`'value1', 'value2', ...`

Are the new values for the updated fields, enclosed in single quotation marks.

`kfield1, kfield2, ...`

Are, at a minimum, all of the key fields for all the segments in the path to the target. Only one target segment is updated. You can include additional fieldname=value pairs for the target or any segment in the path. When additional fieldname=value pairs are present for the target segment, these are used for change verification protocol processing. If all target fieldname=value pairs that are present match the Adabas segment field values, the segment is updated with the SET fieldname values. The field list can contain only elementary fields. Groups, sub-descriptors, super-descriptors, and hyper-descriptors, cannot be used in the field list.

`'kvalue1', 'kvalue2', ...`

Are the values that identify the target segment, enclosed in single quotation marks.

Reference Rules for Updating Records in an Adabas Data Source

- For a segment with a unique key, the key field value and any additional fieldname=value pairs are used to qualify the target segment for an update.
- For a segment with a non-unique key, you must supply the key field. If the WHERE criteria for this type of segment, in the path or target, do not identify a unique occurrence, the first occurrence found is updated. The use of Adabas descriptors for this type of segment is highly recommended for efficiency.
- For a segment with no key, you must supply at least one fieldname=value pair. If the WHERE criteria for this type of segment, in the path or target, do not identify a unique occurrence, the first occurrence found is updated.

Syntax How to Delete Records From an Adabas Data Source

```
DELETE FROM mfname WHERE field1 ='value1' [AND field2 ='value2'...]
```

where:

mfname

Is the name of the Master File to use.

field1, field2, ...

Are, at a minimum, the names of all of the key fields for all the segments in the path to the target. Only one target segment is deleted; Adabas cascades the delete to dependent segments. Additional fieldname=value pairs may be included for the target segment and any segment in the path that has no key or a non-unique key. It is recommended to use Adabas descriptor fields for any additional fieldname=value pairs.

`'value1', 'value2', ...`

Are the values that identify the target segment to be deleted.

Reference Rules for Deleting Records From an Adabas Data Source

- For a segment with a unique key, the key field value is used to delete the target segment.
- For a segment with a non-unique key, you must supply the key field. If the WHERE criteria for this type of segment, in the path or target, do not identify a unique occurrence, the first occurrence found is deleted. The use of Adabas descriptors for this type of segment is highly recommended for efficiency.
- For a segment with no key, you must supply at least one fieldname=value pair. If the WHERE criteria for this type of segment, in the path or target, do not identify a unique occurrence, the first occurrence found is deleted.
- A segment occurrence with ACCESS=PE or MU is deleted from the data source except if it is the last occurrence for an ACCESS=PE segment. Adabas only deletes the last occurrence if all fields have the NU option in the FDT; if they do not all have this option, the occurrence has empty values in all fields.
- When you delete segments that have dependent segments, the DELETED counter for the session may have an incorrect value. For a first level segment with descendants, this counter is always incorrect. For a second level segment, this counter is incorrect if there are multiple descendant segments. For a third level segment, this counter is always correct.

Data Adapter Navigation

To retrieve the necessary records that fulfill a request, the Data Adapter for Adabas navigates the Adabas database through direct calls in read-only mode. The calls generated depend on three factors:

- The information supplied in the Master File.
- The information supplied in the Access File for a specified segment.
- The particulars of the report request.

The data adapter uses information from these sources to choose the most appropriate and efficient retrieval method.

There are three logical types of Adabas access:

- Entry segment retrieval of Adabas records.
- Retrieval of descendant periodic groups and multi-value fields.
- Retrieval of descendant Adabas records.

Entry Segment Retrieval of Adabas Records

The server constructs an Adabas request based on the Master File, Access File, and the report request. The root of the subtree is the entry segment into the database for a particular request.

For Adabas structures, the root of the subtree must be a segment in the Access File containing singly occurring fields (ACCESS=ADBS). The Adabas calls generated to retrieve data for this entry segment depend on the Access File information that has been supplied for the segment and the selection tests in the request.

The first decision the Data Adapter for Adabas makes for the entry segment is whether to retrieve all the records of the segment or just a subset.

All records are retrieved for a segment if either of the following conditions exists:

- There is *no* selection test on any field that is a descriptor (defined in the Access File with TYPE=SPR, DSC, or NOP) and the SEQFIELD has not been defined.
- The specified selection test is *not* IF or WHERE, and does not use the relational operators:

IS, EQ, GE, GT, LT, LE, or FROM...TO

For example:

```
field IS value1 [(OR value2... OR valuen)]
```

```
field IS (ddname)
```

```
field FROM value1 TO value2 [(OR value3 TO value4...)]
```

```
field GE value1
```

```
field GT value1
```

```
field LT value1
```

```
field LE value1
```

```
group EQ 'value1/value2/value3'
```

When you are using one or more of these selection tests against a descriptor, the data adapter uses the inverted list to retrieve a subset of the records for the segment.

For example, consider the following selection criterion:

```
field IS value1 [(OR value2... OR valuen)]
```

In this example, any combination of full values or partial values triggers the use of inverted lists except when the \$ (used to indicate a masked field) is in the first position of the value. If the \$ is in the first position of the value, the search conducted by Adabas is not done through the descriptor's inverted list. Instead, the search is conducted physically through the database in the same manner that a Read Physical call performs a retrieval. See *Read Physical Calls* on page 2-96 for more information.

Once the data adapter determines whether to retrieve all or a subset of the segment records, it decides which access strategy to use based on the Access File attribute settings. The following are the strategies that the adapter can use:

- Retrieval of all records through Read Physical (L2) calls discussed in *Read Physical Calls* on page 2-96.
- Retrieval of all records through Read Logical (L3) calls without using a starting value discussed in *Read Logical Navigation* on page 2-97.
- Retrieval of a subset of records through Read Logical (L3) calls using a starting value discussed in *Read Logical Navigation* on page 2-97.
- Retrieval of all records for the entry segment through Read Logical by ISN (L1) calls discussed in *Read Logical Navigation* on page 2-97.
- Retrieval of a subset of records through simple FIND (S1) calls discussed in *FIND Navigation* on page 2-102.
- Retrieval of a subset of records through complex FIND (S1) calls discussed in *FIND Navigation* on page 2-102.
- Retrieval of all records through Read Descriptor Value (L9) direct calls discussed in *Read Descriptor Value (L9) Direct Calls* on page 2-104.

The selection logic is designed to implement as many of the record selection tests as possible at the Adabas level. It minimizes the number of I/O operations required to access the necessary data by issuing efficient calls to Adabas.

Read Physical Calls

Read Physical calls read every database record in a physical file, and then return every record for the entry segment to the server. The server must then select the records that meet the request's selection criteria and discard the rest.

The Data Adapter for Adabas issues Read Physical (L2) calls to retrieve all records for the entry segment when:

- The request contains no optimizable selection tests on an inverted list (descriptor) and the Access File does not have a SEQFIELD defined.

- With CALLTYPE= FIND, there is no screening on a descriptor, the screening for a non-descriptor has been set off, and the Access File does not have a SEQFIELD defined.

See *Adabas Reporting Considerations* on page 2-78 for more information about retrieval of records with no screening on a descriptor.

The steps the adapter and Adabas perform to complete a Read Physical call are:

1. The adapter constructs a Read Physical (L2) call.
2. Adabas returns each record in the physical file corresponding to the Adabas file number specified. The records are retrieved in the order in which they are physically stored.

Subsequent retrieval for the entry segment is completed by issuing the same Read Physical call with the User Control Block unmodified. The adapter terminates retrieval when one of the following occurs:

- Adabas issues a response code indicating end-of-list.
- The RECORDLIMIT or READLIMIT is reached (see *Adabas Reporting Considerations* on page 2-78 for information about these keywords).

Read Physical (L2) calls can be faster than Read Logical (L3) calls depending on the request and the data dispersion of the physical storage. Ask your Adabas database administrator if you should use SEQFIELD for a given Adabas record, or if Read Physical would be more efficient.

Read Logical Navigation

When designing a database, it is necessary to know the contents of the files being created. Defining descriptors is very important for efficiency. The Data Adapter for Adabas does not have information about the different fields, or about which field is more efficient to parse first, unless the Access File has this information. The Access File tells the adapter how to access your data in the most efficient manner. The order of the fields (for example, DSC, NOP, or SPR descriptors) is very important in the Access File, as the adapter uses this order when executing an IF or WHERE statement.

The Access File needs to have information about the index fields used in the report request. The order in which you define these fields is very important. Adabas is field oriented, not positional. The order of the fields will not affect retrieval, except for the selection of which inverted list to use.

The Access File controls the order of the IF or WHERE statements, so the database administrator can set up the Access File in the order of the most efficient reads. The user will then receive the requested data much more efficiently. The order in the Access File controls the order of retrieval.

Read Logical Calls Without a Starting Value

The Data Adapter for Adabas uses Read Logical (L3) calls to retrieve all records for the entry segment when the Access File contains a SEQFIELD value for that segment and the report request does not contain an optimizable selection test on an inverted list, and CALLTYPE=RL.

SEQFIELD is generally used to suppress Read Physical calls when there is no optimizable selection test on an inverted list. A Read Physical call will be issued if a SEQFIELD is not defined.

The steps the adapter and Adabas perform to complete a Read Logical call without starting values are:

1. The adapter constructs a Read Logical (L3) call without the 'Value Start' option and without the Adabas Search and Value buffers.
2. Adabas returns each record corresponding to an entry in the inverted list. The records are in the same order as the inverted list.

Subsequent retrieval for the entry segment is done by issuing the same Read Logical call with the User Control Block unmodified. The records are returned in ascending value of the inverted list. The adapter terminates retrieval when one of the following occurs:

- Adabas issues a response code indicating end-of-list.
- The RECORDLIMIT or READLIMIT is reached.

In an Adabas file that has no record types, you can select the value of SEQFIELD from any inverted list containing entries for all records on the file.

SEQFIELD is required when the Adabas file has several record types. In this case, the value of SEQFIELD is an inverted list which has entries for a single record type.

If the descriptor used as the SEQFIELD is null-suppressed or contains null-suppressed fields, only records where the descriptor is populated will be included in the retrieval. An entry is not included in the inverted list if a field is null and will not be returned by Adabas to the server. See your Software AG documentation for more information about null-suppression and how it affects data retrieval.

Read Logical Calls With a Starting Value

The Data Adapter for Adabas constructs Read Logical (L3) calls for the root of the accessed subtree when the following conditions exist:

- The Access File contains CALLTYPE=RL for that segment.
- The request contains an optimizable selection test against a descriptor or superdescriptor identified by TYPE=SPR or DSC in the Access File.
- None of the selection tests is on a field identified by TYPE=NOP in the Access File.

When a report request contains multiple optimizable selection tests, the order of descriptors in the Access File determines the order in which the server applies the selection tests. The server issues a Read Logical (RL) call using the first descriptor listed in the Access File that participates in a selection test.

Therefore, for efficient processing, you should describe the most restrictive descriptor at the beginning of its segment in the Access File. The order of descriptors in the Master File has no effect on selection processing.

The steps Adabas performs to complete a Read Logical call are:

1. The adapter constructs a Read Logical (L3) call with the starting value to be retrieved for that inverted list.
2. The adapter calls Adabas with a 'V' in Command Option 2, which instructs Adabas to use the 'Value Start' option.
3. The call retrieves the first record whose value in the inverted list is equal to or greater than the value specified in the request.

Subsequent retrieval for the entry segment is done by issuing the same Read Logical call with the User Control Block unmodified. The records are returned in ascending value of the inverted list.

The adapter terminates retrieval when one of the following occurs:

- The value of the inverted list is out of range of the request.
- Adabas issues a response code indicating end-of-list.
- The RECORDLIMIT or READLIMIT is reached.

If there are multiple values specified in the request (for example, *WHERE field EQ value OR value...*, *WHERE field FROM val1 TO val2 OR val3 TO val4*) and RECORDLIMIT or READLIMIT has not been reached, the Data Adapter for Adabas releases the Command ID and reissues the Read Logical call with the next starting value. For each set of values, the adapter terminates retrieval when one of the above occurs.

Retrieval Through Read Logical by ISN (L1) Calls

The Data Adapter for Adabas uses Read Logical by ISN (L1) calls to retrieve all records for the entry segment when:

- The report request does not contain an optimizable selection test on an inverted list or an ISN list.
- The Access File contains a SEQFIELD value for that segment and this value is equal to the ISN field name.

Adabas returns each record corresponding to an entry in the Address Converter. The records are in ascending value of the ISN. For example:

Access File ADATEST

```
SEGNAM=S01, ACCESS=ADBS, ... , SEQFIELD=ISN_FIELD , $
```

Request

```
SELECT AA_FIELD, AJ_FIELD FROM ADATEST;
```

The Adabas Interface uses Read Logical by ISN (L1) calls to retrieve a single record for the entry segment when:

- The report request contains the only EQ relational operator in the selection test on an ISN list.
- The ISN field is used as the cross-referenced field in the Join to Adabas file.

Adabas returns RC 113 if the record with the ISN defined in the test is not present in the Address Converter for the file. Then the Adabas Interface returns the answer "Record is not found".

For example:

```
SELECT AA_FIELD, AJ_FIELD FROM ADATEST  
WHERE ISN_FIELD = 1100;
```

Note: The Multifetch option will be suppressed for this call.

The Adabas Interface uses Read Logical by ISN (L1) calls to retrieve subset of records for the entry segment when:

- The report request contains the only GE/GT relational operator in the selection test on an ISN list.
- The report request does not contain any selection test on an inverted list.
- The Access File contains a SEQFIELD value for that segment.

Adabas returns each record corresponding to an entry in the Address Converter. The records are in ascending value of the ISN.

For example:

AFD ADATEST contains:

```
SEGNAM=S01, ACCESS=ADBS , ... ,SEQFIELD=AA_FIELD , $
```

Request 1. When Read Logical by ISN (L1) used:

```
SELECT AA_FIELD, AE_FIELD FROM ADATEST  
WHERE ISN_FIELD > 1100;
```

Request 2. When Read Logical (L3) used:

```
SELECT AA_FIELD, AE_FIELD FROM ADATEST
WHERE ISN_FIELD > 1100 AND AJ_FIELD = 'TAMPA';
```

After issuing an Insert request to the file with the ISN field in the MFD, the resulting ISN from Adabas is printed in the message FOC4592:

```
(FOC4592) RECORD IS INSERTED WITH ISN : nnnnn
```

For example:

```
SQL
INSERT INTO ADBTEST (AA_FIELD, AJ_FIELD)
VALUES ('11111111', 'TAMPA');
END

ADBTEST ADBSINX ON 09/20/2002 AT 15.22.16
(FOC4592) RECORD IS INSERTED WITH ISN : 11
(FOC4566) RECORDS AFFECTED DURING CURRENT REQUEST : 1/INSERT
TRANSACTIONS: TOTAL = 1 ACCEPTED= 1 REJECTED= 0
SEGMENTS: INPUT = 1 UPDATED = 0 DELETED = 0
```

If Insert request contains an ISN field in the fields list and a corresponding value is not 0, then the Adabas Interface issues an N2 call to Adabas. Adabas assigns this ISN value to the record provided by the user. Adabas returns RC 113 if this value was already assigned to another record in the file or if it is larger than the MAXISN in effect for the file.

For example:

Request 1.

```
SQL
INSERT INTO ADBTEST (AA_FIELD, AJ_FIELD, ISN_FIELD)
VALUES ('11111114', 'TAMPA', 14);
END

ADBTEST ADBSINX ON 09/20/2002 AT 15.22.16
(FOC4592) RECORD IS INSERTED WITH ISN : 14
(FOC4566) RECORDS AFFECTED DURING CURRENT REQUEST : 1/INSERT
TRANSACTIONS: TOTAL = 1 ACCEPTED= 1 REJECTED= 0
SEGMENTS: INPUT = 1 UPDATED = 0 DELETED = 0
```

Request 2.

```
SQL
INSERT INTO ADBTEST (AA_FIELD, AJ_FIELD, ISN_FIELD)
VALUES ('11111114', 'TAMPA', 999999);
END

ADBEMP43ADBSIN ON 09/19/2002 AT 16.16.22
(FOC4561) ERROR IN ADABAS INCLUDE /113
TRANSACTIONS: TOTAL = 1 ACCEPTED= 1 REJECTED= 0
SEGMENTS: INPUT = 0 UPDATED = 0 DELETED = 0
```

FIND Navigation

The manipulation of ISN lists is done on the initial call to Adabas. Both the intermediate lists and the resulting list may be very large and may take up a large percentage of Adabas workspace. The Adabas work area is shared by all Adabas users, and complex FINDs may affect programs that are updating the database. You may want to suppress FINDs altogether. Accomplish this suppression by specifying CALLTYPE=RL on every Access File segment. Ask your Adabas database administrator whether to use FIND processing.

Simple FIND Calls

The Data Adapter for Adabas constructs a simple FIND (S1) call, using a single inverted list structure, for the root of the accessed subtree if one of the following conditions is met:

- The Access File contains either CALLTYPE=FIND for that segment or doesn't specify the CALLTYPE attribute, in which case the adapter default is CALLTYPE=FIND. Additionally, the request contains a single selection test on a descriptor identified with TYPE=DSC or a superdescriptor identified with TYPE=SPR in the Access File.
- For any value of CALLTYPE (RL or FIND), if the request contains a single selection test on a subdescriptor or superdescriptor containing partial fields (identified with TYPE=NOP in the Access File).

CALLTYPE=FIND instructs the adapter to generate FIND calls to retrieve records from the inverted lists. This method is the default for processing selection criteria on inverted lists for a segment (if CALLTYPE is omitted from the Access File segment declaration).

If you have selected a field defined with TYPE=NOP (for example, sub descriptors), a FIND call is issued even if RL has been specified in cases when:

- SET NOPACCESS FIND was performed.
- The server is running in an OpenVMS environment.
- The same Adabas field is defined in a descendant PE/MU segment.

Switching from RL to FIND mode is also performed if:

- A field is defined with TYPE=SPR and the server is running in an OpenVMS environment.
- A field is defined with TYPE=PDS (phonetic descriptor) or TYPE=HDS (hyper descriptor).
- A field is defined in a PE segment (that it is a part of periodic group).

For performance considerations, a FIND (S1) call does not issue a READ ISN (L1) call if an answer set containing only one record is returned. The GET FIRST option returns the first record in the inverted list. This reduces unnecessary I/O calls. If the FIND call returns more than one ISN in a list, the GET NEXT option of L1 is used to start reading the ISN list from the second entry.

The steps Adabas performs to complete a FIND (S1) call are:

1. The adapter constructs a FIND (S1) call with the value(s) specified.
2. The adapter calls Adabas with an 'H' in Command Option 1, which instructs Adabas to store the resulting list of Internal Sequence Numbers (ISNs) in the Adabas work area.
3. Adabas constructs a complete list of ISNs for every record matching the selection criteria in the work area of Adabas. This list is sorted in ascending ISN order.
4. The adapter issues a Read ISN (L1) call to retrieve the first record from the Adabas work area which matches the selection criteria. The records are returned in ascending ISN order.

All subsequent retrieval for this entry segment is done using the Read ISN (L1) call issued against the ISN list held by Adabas. L1 commands are issued until one of the following occurs:

- Adabas issues a response code indicating end-of-list.
- The RECORDLIMIT or READLIMIT is reached.

Complex FIND Calls

The Data Adapter for Adabas constructs complex FIND (S1) calls, using two or more inverted list structures, for the root of the accessed subtree if one of the following conditions is met:

- The Access File contains CALLTYPE=FIND for that segment (or omits CALLTYPE). The request contains multiple selection tests on a descriptor or superdescriptor described with TYPE=DSC or TYPE=SPR in the Access File.
- For any value of CALLTYPE, the request contains selection tests on a subdescriptor or superdescriptor containing partial fields (identified with TYPE=NOP in the Access File).

The steps Adabas performs to complete a FIND call on multiple inverted list structures are:

1. The adapter constructs a FIND (S1) call with all the values specified for each inverted list on which there are selection criteria.
2. The adapter calls Adabas with an 'H' in Command Option 1, which instructs Adabas to store the resulting list of ISNs in the Adabas work area.
3. Adabas constructs an ISN list for each inverted list specified.
4. Adabas merges these lists into a final list of ISNs which match all of the selection criteria in the call. This list is kept in the Adabas work area and is sorted in ascending ISN order.
5. The adapter then issues a Read ISN (L1) call to retrieve the first record from the list in the Adabas work area. The records are returned in ascending ISN order.

All subsequent retrieval for this entry segment is completed using Read ISN (L1) calls issued against the ISN list held by Adabas. L1 commands are issued until one of the following occurs:

- Adabas issues a response code indicating end-of-list.
- The RECORDLIMIT or READLIMIT is reached.

Read Descriptor Value (L9) Direct Calls

The Data Adapter for Adabas issues the Adabas Read Descriptor Value (L9) direct call when you use the COUNT command or SUM CNT.*field* within a report request. L9 calls are, by definition, limited to descriptors defined to the server with the following:

- INDEX=I in the Master File.
- A field type of NOP, DSC, or SPR (defined in the Access File).

L9 calls allow an aggregate ISN count to be returned for each unique value on the inverted list at a cost of only one call per unique value. This technique allows a dramatic efficiency gain over passing each record to the server to process the count.

Selection criteria (for example, WHERE *descriptor* EQ '1234') are passed along with the L9 call to further limit the number of calls. If any non-descriptor fields are mentioned in the TABLE request, a Read Descriptor Value (L9) direct call is not possible and the server uses its own internal count processing. If a BY field is used, it must be the same field or GROUP name used as the object of the COUNT verb.

Here is an example of the SUM CNT.*field* using EMPFILE1:

```
TABLE FILE EMPFILE1
SUM CNT.DEPT_S03
BY DEPT_S03
END
```


Descendant Periodic Groups and Multi-value Fields

The Data Adapter for Adabas must retrieve a count of the number of occurrences before attempting to retrieve a periodic group (PE) or multi-value (MU) field. This retrieval is accomplished by taking the appropriate Adabas counter field (indicated by the OCCURS attribute on the segment declaration in the Master File) and placing its ALIAS (the two-character Adabas name of the PE or MU appended with the letter C) in the Format buffer of the call for the parent record fields.

Adabas returns to this counter field the total number of occurrences that exist for the PE or MU.

The adapter retrieves the descendant data in one of the following ways:

- For MU fields or PE groups which do not contain an MU, all the occurrences are retrieved at once. The counter field ALIAS is used without the letter C, to retrieve the entire set of occurrences in one Read ISN (L2) call.
- For PE groups which contain one or more MU fields, the adapter retrieves a single occurrence of all component fields at a time, including the count(s) of the number of MU descendants, if required. For example, a PE group containing an MU field which has five occurrences of the PE requires five Read ISN (L2) calls to retrieve all of them.

Descendant Adabas Records

The access strategy used to retrieve descendant records in a subtree depends on the Access File attributes specified for a given segment and the SEGTYPE attribute in the Master File (or the SEGTYPE created on the cross-referenced segment as the result of a JOIN). In some cases, the selection criteria in a request further affect access strategy for descendant segments.

Descendant records are related to their parents by a field or GROUP in the parent that corresponds to a field or GROUP in the descendant. This relationship is set up either in the Access File using the KEYFLD/IXFLD pair or through the host field/cross-referenced field combination in a JOIN.

The most appropriate access strategy is selected based on the value of CALLTYPE of the descendent segment in the Access File. The three basic strategies used to obtain descendant records are:

- Retrieval of descendant records through Read Logical calls using a starting value.
- Retrieval of descendant records through simple FIND calls.
- Retrieval of descendant records through complex FIND calls.

The CALLTYPE attribute, the KEYFLD/IXFLD attributes, the SEGTYPE, and the TABLE request are the determining factors in the way descendant data is retrieved.

Read Logical Calls Using a Starting Value

The Data Adapter for Adabas constructs Read Logical (L3) calls for related descendant records of a parent when the following conditions are met:

- The Access File contains CALLTYPE=RL for that segment.
- The IXFLD or JOIN to field is described as TYPE=DSC or TYPE=SPR in the Access File.
- No other selection criteria on inverted lists for this segment have selection on a descriptor with TYPE=NOP.

Read Logical processing is performed on a single inverted list only. The steps the adapter performs to complete a Read Logical call using a starting value are:

1. The adapter constructs a Read Logical (L3) call with the value of the KEYFLD from the parent segment for the IXFLD inverted list. This call retrieves the first record whose value in the inverted list is equal to or greater than the value of the KEYFLD.
2. For unique descendants, no additional retrieval is performed on the descendant segment for any given parent. (For an embedded cross-referenced segment, the SEGTYPE is U or KU or the segment was the cross-referenced segment in a unique JOIN.)
3. For non-unique descendants, subsequent retrieval for the entry segment is completed by issuing the same Read Logical call with the User Control Block unmodified. The records are returned in ascending value of the inverted list.

The adapter terminates retrieval when one of the following conditions is met:

- The value of the inverted list is greater than the value of the KEYFLD.
- Adabas issues a response code indicating end-of-list.
- The RECORDLIMIT or READLIMIT is reached.

Simple FIND Calls (Descendant Records)

The Data Adapter for Adabas constructs a simple FIND (S1) call, using a single inverted list, to obtain related descendant records for a parent when one of the following conditions exists:

- The Access File contains CALLTYPE=FIND (or omits CALLTYPE) and the request contains no selection criteria on the descendant segment.
- The Access File contains CALLTYPE=FIND (or omits CALLTYPE) and the request contains selection criteria on the descendant segment and the descendant is unique. (For an embedded cross-referenced segment, the SEGTYPE is U or KU or the segment was the cross-referenced segment in a unique JOIN.)
- For any value of CALLTYPE, IXFLD is described with TYPE=NOP in the Access File and the request contains no selection criteria on the descendant segment.

The steps Adabas performs to complete simple FIND calls against descendant segments are:

1. The adapter constructs a FIND (S1) call with the value of the KEYFLD from the parent segment for the inverted IXFLD list.
2. For unique descendants, Adabas returns the record on the FIND call; no 'H' command is issued to Adabas. Only one call will be issued, and only the first instance of data will be returned.
3. For non-unique descendants, the adapter calls Adabas with an 'H' in Command Option 1, which instructs Adabas to store the resulting list of ISNs in the Adabas work area.
4. Adabas constructs a complete list of ISNs for every record related to the parent record in the work area. This list is sorted in ascending ISN order.
5. The adapter issues a Read ISN (L1) call to retrieve the first record from the Adabas work area which matches the selection criteria.

If the descendant is not unique, subsequent retrieval of records for this descendant segment is completed using the Read ISN (L1) call issued against the ISN list held by Adabas. L1 calls are issued until one of the following occurs:

- Adabas issues a response code indicating end-of-list.
- The RECORDLIMIT or READLIMIT is reached.

Complex FIND Calls (Descendant Records)

The Data Adapter for Adabas constructs a complex FIND (S1) call, using several inverted list structures, to obtain related descendant records for a parent when one of the following conditions is met:

- The Access File contains CALLTYPE=FIND for that segment (or omits CALLTYPE) and the descendant is non-unique. (For an embedded cross-referenced segment, the SEGTYPE is S or KM, or the segment was the cross-referenced segment in a non-unique JOIN.) The report request contains one or more selection criteria on the descendant segment.
- The Access File contains any value of CALLTYPE. The descendant is non-unique. (For an embedded cross-referenced segment, the SEGTYPE is S or KM, or the segment was the cross-referenced segment in a non-unique JOIN.) The request contains one or more selection criteria on the descendant segment, one of which is on a descriptor, or the IXFLD is described as TYPE=NOP in the Access File.

The steps Adabas performs to complete a complex FIND call are:

- 1.** The adapter constructs a FIND (S1) call with the value of the KEYFLD from the parent segment using the inverted IXFLD list, in addition to all other selection criteria on inverted lists from the request.
- 2.** For non-unique descendants, the adapter calls Adabas with an 'H' in Command Option 1, which instructs Adabas to store the resulting list of ISNs in the Adabas work area.
- 3.** Adabas constructs an ISN list for each inverted list specified, and merges these lists into a final list of ISNs that match all of the selection criteria in the call. This list is kept in the Adabas work area and is sorted in ascending ISN order.
- 4.** The adapter then issues a Read ISN (L1) call to retrieve the first record from the Adabas work area which matches the selection criteria.

Subsequent retrieval of records for this descendant segment is done through the Read ISN (L1) call issued against the ISN list held by Adabas. L1 commands are issued until one of the following occurs:

- Adabas issues a response code indicating end-of-list.
- The RECORDLIMIT or READLIMIT is reached.

CHAPTER 3

Getting Started in Allbase

Topics:

- Preparing the Allbase Environment
- Configuring the Data Adapter for Allbase
- Managing Allbase Metadata
- Customizing the Allbase Environment

The Data Adapter for Allbase allows applications to access Allbase data sources. The adapter converts application requests into native Allbase statements and returns optimized answer sets to the requesting application.

Preparing the Allbase Environment

For the Allbase Data Adapter, no environment variables need to be set prior to starting the server.

Configuring the Data Adapter for Allbase

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

There is only one method by which a user can be authenticated when connecting to an Allbase database server: Operating System (Trusted Mode). User ID and password used to log on to the operating system are automatically used to connect to the Allbase database server. The server data access agent impersonates an operating system user according to the server deployment mode. The agent process establishes a connection to an Allbase database server based on the impersonated operating system user credentials.

Syntax How to Declare Connection Attributes Manually

```
ENGINE [SQLALB] SET DATABASE database_name
```

where:

SQLALB

Indicates the Allbase Data Adapter. You can omit this value if you previously issued the SET SQLENGINE command.

database_name

Indicates the Allbase database name used for this connection. This parameter must be supplied.

Note: Parameters containing special characters should be enclosed in single quotation marks.

Example Declaring Connection Attributes Manually

The following SET DATABASE command connects to the Allbase database server named SAMPLESERVER. To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLALB SET DATABASE sampleserver
```

Syntax **How to Set the Home Location Path**

```
ENGINE [SQLALB] SET HOME location_path
```

where:

SQLALB

Indicates the Allbase Data Adapter. You can omit this value if you previously issued the SET SQLENGINE command.

location_path

Indicates the location of the Allbase database used for this connection. This parameter must be supplied.

Note: Parameters containing special characters should be enclosed in single quotation marks.

Example **Setting the Home Location Path**

```
ENGINE SQLALB SET HOME /opt/allbase/mydb
```

Reference **Declaring Allbase Connection Attributes From the Web Console**

Attribute	Description
Data source	Enter the Allbase database name.
Home directory	Enter the location of the Allbase database.

Overriding the Default Connection

Once connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax **How to Change the Default Connection**

```
ENGINE [SQLALB] SET DEFAULT_CONNECTION [connection]
```

where:

SQLALB

Indicates the Data Adapter for Allbase. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the service name defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following error message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following error message is issued: FOC1671, Command out of sequence.

Example **Selecting the Default Connection**

The following SET DEFAULT_CONNECTION command selects the Allbase database server named SAMPLENAME as the default Allbase database server:

```
ENGINE SQLALB SET DEFAULT_CONNECTION SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax **How to Control the Connection Scope**

```
ENGINE [SQLALB] SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

SQLALB

Indicates the Data Adapter for Allbase. You can omit this value if you previously issued the SET SQLENGINE command.

FIN

Disconnects automatically only after the session has been terminated. This value is the default.

COMMIT

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Allbase Metadata

When the server accesses a data source, it needs to know how to interpret the data that it finds. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Allbase data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Allbase table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Editing Metadata page opens.

To create a synonym, you must have configured the adapter. See *Configuring the Data Adapter for Allbase* for more information.

2. Expand the *Add* folder, expand the adapter folder, and then click a connection. The right pane displays table and view selection options:

Select All Tables/Views. Select this radio button to create synonyms for all tables and views. This value is the default.

Filter by Name, Owner and Table Type. Select this radio button to filter the tables or views for which to create synonyms.

Selecting this option adds the following:

The screenshot shows a web console interface with a dark blue background. At the top, there are two radio button options: "Select all Tables/Views" (which is unselected) and "Filter by Name, Owner and Table Type" (which is selected). Below these options, there are three rows of input fields and controls:

- Owner:** A text input field with a sample value "abc%" to its right.
- Table name:** A text input field with a sample value "abc%" to its right. The field contains the text "NF%".
- Table type:** Three radio button options: "TABLE" (selected), "VIEW", and "TABLE/VIEW".

At the bottom of the form, there is a button labeled "Select Tables".

Owner. Enter a string for filtering the owners' IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owners' IDs begin with the letters ABC; %ABC to select tables or views whose owners' IDs end with the letters ABC; %ABC% to select tables or views whose owners' IDs contain the letters ABC at the beginning, middle, or end.

Table Name. Enter a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

Table Type. Select one of the following options: Tables (this is the default), Views, Tables/Views.

3. Click *Select Tables*. All tables that meet the specified criteria are displayed:

Prefix: SET CONVERSION LONGCHAR

Select Application Directory:

All Select

Check	Default Synonym Name	Owner/Schema	Table Name	Type
<input type="checkbox"/>	<input type="text" value="NF29001"/>	EDAQA	NF29001	TABLE
<input type="checkbox"/>	<input type="text" value="NF29002"/>	EDAQA	NF29002	TABLE
<input type="checkbox"/>	<input type="text" value="NF29003"/>	EDAQA	NF29003	TABLE
<input type="checkbox"/>	<input type="text" value="NF29004"/>	EDAQA	NF29004	TABLE
<input type="checkbox"/>	<input type="text" value="NF29005"/>	EDAQA	NF29005	TABLE
<input type="checkbox"/>	<input type="text" value="NF29006"/>	EDAQA	NF29006	TABLE
<input type="checkbox"/>	<input type="text" value="NF29007"/>	EDAQA	NF29007	TABLE
<input type="checkbox"/>	<input type="text" value="NF29008"/>	EDAQA	NF29008	TABLE
<input type="checkbox"/>	<input type="text" value="NF29009"/>	EDAQA	NF29009	TABLE
<input type="checkbox"/>	<input type="text" value="NF29012"/>	EDAQA	NF29012	TABLE
<input type="checkbox"/>	<input type="text" value="NF29013"/>	EDAQA	NF29013	TABLE

4. If you have tables with identical table names, assign a prefix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave field blank.

5. From the SET CONVERSION LONGCHAR drop-down list, select: Text, Alpha, or BLOB. The default value is Text.
6. From the Select Application Directory drop-down list, select a directory. The default value is baseapp.
7. Complete your table or view selection:
To select all tables or views in the list, click **All**.
To select specific tables or views, click the corresponding check boxes.
8. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.
9. Click *Create Synonym*. Synonyms are created and added under the specified application directory.

Reference **Managing Synonyms**

In the navigation pane, click the name of the synonym to access the following options:

Sample Data	Retrieves up to 20 rows from the associated data source.
Properties	Displays a graphic representation of the synonym and enables you to edit its metadata.
Edit Master File	Enables you to manually edit the synonym's Master File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Master File, to update the synonym.
Edit Access File	Enables you to manually edit the synonym's Access File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Access File, to update the synonym.
Refresh	Regenerates the synonym. Use this option if structural changes were made to the data source.
Drop	Deletes the synonym.
Copy to	Copies the synonym to another application directory. Click the target directory from the resulting list.
Move to	Moves the synonym to another application directory. Click the target directory from the resulting list.

Syntax **How to Create a Synonym Manually**

```
CREATE SYNONYM app/synonym FOR table_view DBMS SQLALB
END
```

where:

app

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, then this application name must not be used.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema) as follows:

```
[owner.]table
```

SQLALB

Indicates the Data Adapter for Allbase.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR edaqa.nf29004 DBMS SQLALB  
END
```

Generated Master File nf29004.mas

```
FILE=NF29004 , SUFFIX=SQLALB ,  
SEGNAME=NF29004 , SEGTYPE=S0 ,  
FIELD=DIVISION4 , 'DIVISION4 ' ,  
I11 ,I4 ,MISSING=OFF ,  
FIELD=DIVISION_NA4 , 'DIVISION_NA4 ' ,  
A25 ,A25 ,MISSING=ON ,  
FIELD=DIVISION_HE4 , 'DIVISION_HE4 ' ,  
I11 ,I4 ,MISSING=ON ,
```

Generated Access File nf29004.acx

```
SEGNAME=NF29004 ,  
TABLENAME=EDAQA.NF29004 ,  
CONNECTION=local ,  
KEYS=1  
,
```

Reference Access File Keywords

Keyword	Description
TABLENAME	Identifies the Allbase table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: TABLENAME=[owner.]table

Data Type Support

The following chart provides information about the default mapping of Allbase data types to server data types:

Allbase Data Type	Data Type		Remarks
	Usage	Actual	
CHAR (n)	A(1...3996)	A(1...3996)	
VARCHAR (n)			
DECIMAL(p,s) p in (1...27) s in (0...27) DEC(27,0) ... default value	P14	P29.27	NUMERIC is a synonym for DECIMAL
FLOAT(p) p in (25...53) Float(53) ... default value	D8	D20.2	DOUBLE PRECISION is a synonym for FLOAT(53).
FLOAT(p) OR REAL p(1...24) Float(24) ... default value	D8	D20.2	REAL is a synonym for FLOAT(24).
INTEGER	I4	I11	
SMALLINT	I4	I6	
DATE	A10	A10	
TIME	A8	A8	
DATETIME	A23	A23	
INTERVAL	A20	A20	

Allbase Data Type	Data Type		Remarks
	Usage	Actual	
BINARY(n) in (1...3996) LONG VARBINARY(n) in (2 ^31 - 1) BLOB BLOB	A(1...3996)	A(1...3996)	
VARBINARY(n) in (1...3996)	A(1...3996)	A(1...3996)	
LONG BINARY	BLOB	BLOB	
LONG VARBINARY	BLOB	BLOB	

CLOB Activation

The default mapping for the Allbase data types CHAR and NCHAR is the data type ALPHA. The ALPHA data type maximum supported length is 4096 characters for TABLE/MODIFY and 32768 characters for API applications. It is now possible to perform SELECT, INSERT, and UPDATE on columns with these data types without using the server data type CLOB.

Syntax **How to Set Server CLOB Activation**

To activate the support for the server data type CLOB, you must issue the following command in one of the supported server profiles:

```
ENGINE [SQLALB] SET CONVERSION LONGCHAR TEXT
```

where:

SQLALB

Indicates the Allbase Data Adapter. You can omit this value if you previously issued the SET SQLENGINE command.

TEXT

Activates long character support using data type CLOB. The default is ALPHA.

Customizing the Allbase Environment

The Data Adapter for Allbase provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Syntax **How to Obtain the Number of Rows Updated or Deleted**

```
ENGINE [SQLALB] SET PASSRECS {ON|OFF}
```

where:

SQLALB

Indicates the Data Adapter for Allbase. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. This value is the default.

OFF

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

CHAPTER 4

Getting Started in Bull GCOS

Topics:

- Preparing the Bull GCOS Environment
- Configuring the Data Adapter for Bull GCOS
- Managing Bull GCOS Metadata
- Customizing the Bull GCOS Environment
- Enabling iWay for Bull Run-Time Environment (RTE)

The Data Adapter for Bull GCOS allows applications to access Bull GCOS data sources. The adapter converts application requests into native Bull GCOS statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

The Data Adapter for Bull GCOS is a TN FOCUS Adapter connecting to a Bull GCOS machine. Therefore, throughout this chapter, the Data Adapter for Bull GCOS is also referred to as TN FOCUS.

Preparing the Bull GCOS Environment

Open services (TCP/IP, TELNET, FTP) are provided on Bull machines through the OPEN7 (GCOS7) and MAINWAY, GNSP, or FCP8 (GCOS8) environments depending on the systems configuration. GNSP (GCOS Network Service Processor) and FCP8 are used only on DPS9000 systems.

Operations Considerations

For GCOS7, the OPEN7 environment must be up with the required server daemons (FTP, TELNET, and TNVIP). The TNVIP Server must be up and running and the relevant listening port number must be noticed (the default port number is 7323). For details, see the *OPEN7 Installation and Operating* manuals for details.

Bull XTA systems do not require OPEN7 but they do require FTP7. This facility is supported by adding @xta to the end of a SET FOCPATH command or by creating a connection profile name starting with the "xta" prefix. The Bull XTA/TNVIP Server on NT must be installed and properly configured.

For GCOS8, the TNVIP gateway must be up and running and the relevant listening port number must be noticed. FTP services may be provided by the same gateway server (GNSP) or by a different one (FCP8). When the FTP services are provided separately from the TNVIP services a SET FTPSERVER command must be used to redefine the FTP Server.

Configuration Considerations

For GCOS7, the access to the Bull GCOS environment is given using the TNVIP Server. A connection profile must be configured for each connecting user to the iWay for Bull Run-Time Environment (RTE). The defined TNVIP connection profile name will be used by the iWay Server when connecting to the iWay for Bull RTE. For details, see the *TNVIP Server Administrator's* manual.

When the TNVIP Server is configured on a gateway platform (UNIX/NT) the GCOS7 FTPSERVER setting is required to allow the data transfer between the Bull GCOS7 and the iWay server which may be different from the TNVIP gateway platform. In this situation, use the SET FTPSERVER command to redefine the FTP Server location.

Important:

1. The iWay for Bull RTE sharable module must be preloaded into the Backing store area (BKST). For details, see the *iWay for Bull RTE Operations and Installation* manual.
2. The iWay for Bull RTE must be started at the time of user connection using the appropriate working directory (library) FOCSLIB parameter, the FOCUS profile FOCEXEC file containing the FILEDEF settings for all of the Bull GCOS DBMS files accessed during the session, and the USERPATH setting for the required iWay for Bull RTE files (Master File and Access File).

3. The AUTODISCONNECT setting will keep the loaded and running FOCUS session active during the server lifetime. The server's previously created user context will be kept and reused by the adapter to communicate with the iWay for Bull RTE session until you reset the AUTODISCONNECT option or end the server session. When set, AUTODISCONNECT locks the user context and rejects any further user context modification.

The GCOS8 platform is accessed through the MAINWAY gateway or the GNSP gateway, which provides the TNVIP services. The MAINWAY or GNSP administrator must configure the connection profile in order to enable access to iWay for Bull RTE. The created connection profile must be prefixed by either 'TSS' or 'FSO' and must have at least 4 characters (TSS0, FSO3, TSS01, etc.).

If additional connection parameters have to be provided to start the iWay for Bull RTE session on the Bull mainframe, a connection script file (tnfocus.scr) must be created and stored in the \$EDACONF/etc directory. This script file must contain on each line the dialogue pattern identification followed by an equal sign (=) and the response to be sent to the Bull mainframe.

For example, the content of the tnfocus.scr file could be:

```
$$ 4200 MODEL=DKU7211
$*$=CN MB FOCUS SC BC41
```

If Bull MAINWAY is used, the FTP protocol is done through a specific gateway called FCP8, which has its own IP address. This FTP Server name must be introduced by using the ENGINE TNFOCUS SET FTPSERVER command.

GNSP gateway provides both services (TELNET and FTP) so you do not need to redefine a FTP Server.

Functional Limitation Considerations

The Data Adapter for Bull GCOS has the following functional limitations:

- JOIN is supported using SQL access on intermediary file.
- Multi-path query is not supported.
- Aggregation, SET, COMPUTE and DEFINE expressions are not passed to the TNFOCUS adapter but are performed by iWay on UNIX/NT.
- UFAS and IDS2 DBMS Data access drivers have the functional limitation level of iWay for Bull RTE version 6.7.1. Please refer to the *iWay for Bull RTE* documentation for more details.

Configuring the Data Adapter for Bull GCOS

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Procedure How to Configure the Adapter Using the Web Console

Before you begin this configuration process, you must start the Web Console.

1. From the left panel of the Web Console, select *Data Adapters*.
2. In the left panel, expand Add and select *BULL GCOS*.
3. Click *BULL GCOS* to open the following window:

Add BULL GCOS to Configuration

HOSTNAME

Security:

Explicit Trusted

FOCLIB

FTP Server

FTP User

FTP Password

On Changing the Configuration, All the Running Agents Will be Killed.

License Code:

Environment:
Must be defined before the server starts

Provide the valid values of all input fields and click *Configure* to update the *edasprof.prf* file with configured adapter connection attributes.

4. Once the adapter is configured, the connection name appears in the left panel under the expanded Configured file, as shown below:



If you left click the connection name, the available options are displayed as shown above.

Reference Configuration Fields

The following is a summary of the Data Adapter for Bull GCOS configuration:

Field	Description
FTP Server	The FTP Server name or IP address.
HOSTNAME	The TNVIP Server name or IP address and service. The # sign introduces the TNVIP service port number. The syntax is: <code>192.168.4.182#7323</code> or <code>hostname#tnvip</code>
FTP User	The FTP user ID for accessing Bull GCOS.
FTP Password	The FTP password for accessing Bull GCOS.
Security	The Bull GCOS access rights definition (user ID and password). Explicit must be checked for Bull GCOS access rights setup.
FOCLIB	The iWay for Bull RTE working directory path definition and user connection profile name. The @ sign introduces the TNVIP user defined connection profile. See example given below for GCOS8 access. <code>'focus/qa/fso/users@fso00'</code>

Declaring Connection Attributes

You can use the SET USER command or the SET CONNECTION_ATTRIBUTES command to set parameters for connecting to the iWay for Bull RTE. This command can be stored in the user profile (for example, EDASPROF.PRF) or in a FOCEXEC procedure.

Syntax

How to Declare Connection Attributes Using SET CONNECTION_ATTRIBUTES

```
ENGINE TNFOCUS SET CONNECTION_ATTRIBUTES text
```

where:

TNFOCUS

Indicates the Data Adapter for Bull GCOS data source.

text

Defines the iWay for Bull RTE host name or IP address, followed by the port number, the user ID and password used to connect, and the library name (iWay for Bull RTE working directory) on Bull GCOS where the data is located.

Example

Declaring Connection Attributes Using SET CONNECTION_ATTRIBUTES

```
ENGINE TNFOCUS SET CONNECTION_ATTRIBUTES
192.168.4.241#7323/AB1,11AB:FOCUS,AB.SL1@F0001
```

where:

AB1

Is the user ID.

11AB

Is the password.

FOCUS.AB.SL1

Is the iWay for Bull RTE working directory (FOCSLIB setting).

F0001

Is the TNVIP gateway connection profile to the iWay for Bull RTE. In this case, 192.168.4.241 is the Bull host IP address, and 7323 is the TNVIP gateway service address.

Reference **Naming Conventions for the Connection Profile**

Following are naming conventions for the connection profile:

- The connection profile name may not exceed five (5) alphanumeric characters and the first character cannot be a numeric character.
- The connection profile name prefixed by FSO or TSS are reserved for GCOS8. For example, FOC01 (GCOS7) and TSS01 (GCOS8) are valid profiles. XTA prefix is reserved for XTA platforms connection profiles.

Bull GCOS7 and GCOS8 file systems are different and the above example is based on the GCOS7 file system syntax. The directory (library) setting can be overridden by the FOCPATH setting. This is useful when the directory (library) path is too long to fit within the SET USER command line. The FOCPATH setting must follow a SET CONNECTION_ATTRIBUTES setting having an incomplete directory path in order to set the proper directory path of the remote iWay for Bull RTE. The directory path must be quote protected for GCOS8 and GCOS7/XTA systems. For example, if the target iWay for Bull RTE working directory is the following:

```
FOCUS/QA/FSO/WKS1USERS/FOCUS
```

The SET CONNECTION_ATTRIBUTES command may be set as:

```
ENGINE TNFOCUS SET CONNECTION_ATTRIBUTES  
192.168.4.241#7323/AB1,11AB:`FOCUS/QA@F0001`
```

or

```
ENGINE TNFOCUS SET CONNECTION_ATTRIBUTES  
192.168.4.241#7323/AB1,11AB:FOCUS@F0001
```

The next FOCPATH setting will properly set the path:

```
ENGINE TNFOCUS SET FOCPATH 'FOCUS/QA/FSO/WKS1USERS/FOCUS'
```

The incomplete directory path given in the SET CONNECTION_ATTRIBUTES command must be quote protected if it contains a '/' (subdirectory separation). In private deployment mode and in order to allow multiple users to access the same Bull Server, multiple user profiles must be created in the \$EDACONF/etc directory. For example, if user1 and user2 are both connecting to server dps7, you must have user1.prf and user2.prf.

Content of user1.prf:

```
ENGINE TNFOCUS SET CONNECTION_ATTRIBUTES  
192.168.4.241#7323/user1,pass1:FOCUS.USER1.SL@foc01
```

Content of user2.prf:

```
ENGINE TNFOCUS SET CONNECTION_ATTRIBUTES  
192.168.4.241#7323/user2,pass2:FOCUS.USER2.SL@foc02
```

Note: Defined user context (access rights, user ID/password, the iWay for Bull RTE working directory (library), and the connection profile must be previously set by the Bull Data and Network Administrator.

Controlling the Connection Scope

Setting AUTODISCONNECT to 'ON' or '1' implies that the iWay for Bull GCOS Run-Time Environment (RTE) session is not disconnected at the end of the request so the subsequent request will not need to start from the connection step. If the value is 'OFF' or '0', each request will start by newly connecting to the iWay for Bull GCOS RTE. The default value is OFF which means the session is disconnected when the request processing is done. For more information, see *Enabling iWay for Bull Run-Time Environment (RTE)* on page 4-17.

Syntax How to Control the Connection Scope

```
ENGINE TNFOCUS SET AUTODISCONNECT x
```

where:

x

Is one of the following values:

0 or OFF. When set to 0 or OFF, the iWay for Bull GCOS RTE session is disconnected at end of the request.

1 or ON. When set to 1 or ON, the iWay for Bull GCOS RTE session is not disconnected at the end of the request.

When this command is used with value 1, the following message appears:

```
(FOC43523) TNFOCUS SESSION AUTODISCONNECT OPTION IS SET: 1
```

When this command is used with value 0, the following message appears:

```
(FOC43523) TNFOCUS SESSION AUTODISCONNECT OPTION IS UNSET
```

Important: This setting is active only in a pooled deployment of the server and when set, the user context cannot be modified.

Managing Bull GCOS Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Bull GCOS data types.

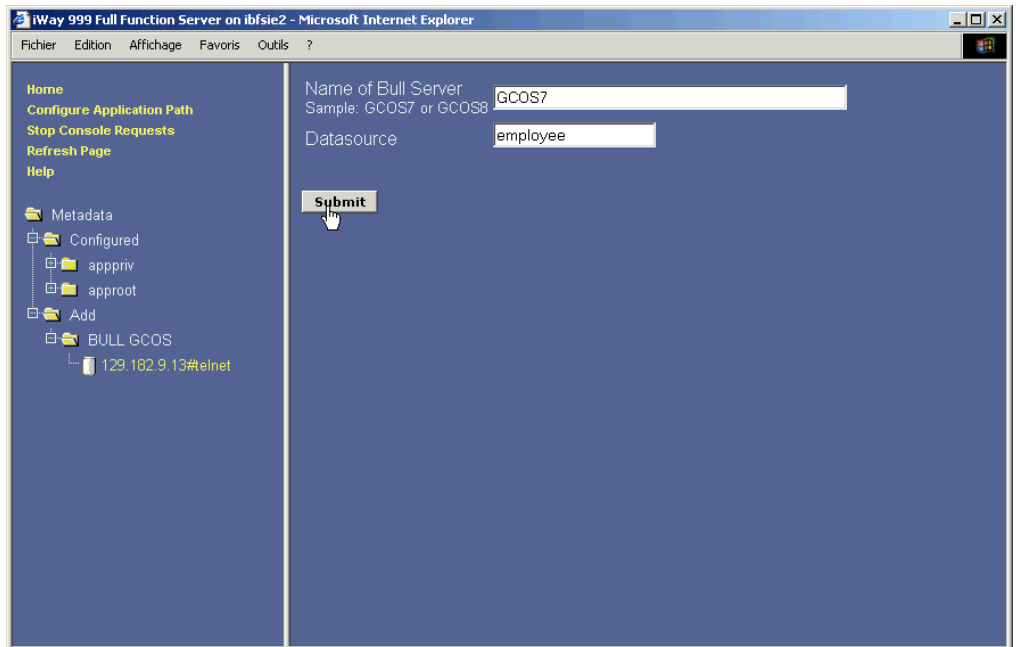
Creating Synonyms

Synonyms define unique names (or aliases) for each TNFOCUS view that is accessible from a server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

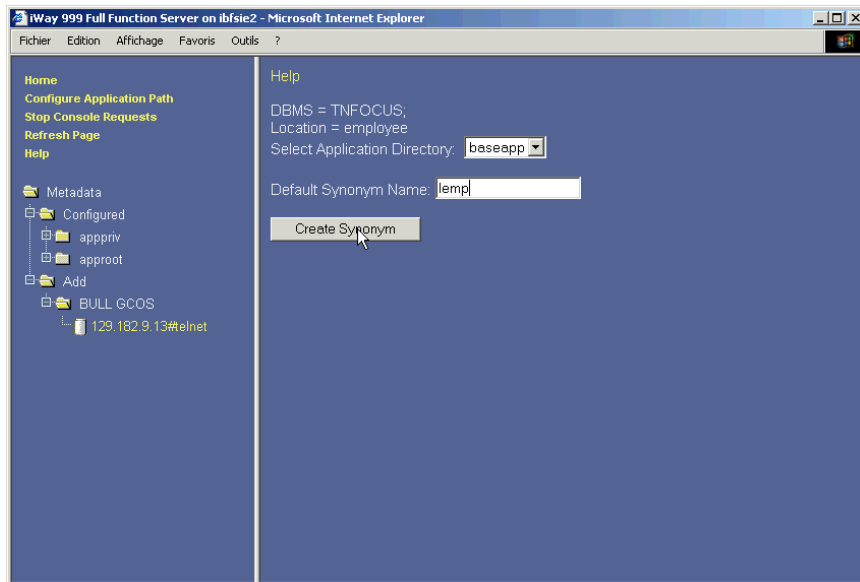
Procedure How to Create a Synonym Using the Web Console

1. Start the Web Console. In the left pane of the Web Console, select *Metadata*. The following window opens:



In the left panel, expand the *Add* folder and the adapter folder (in this example, *BULL GCOS*), and click the configured connection name to enter the Create Synonym window.

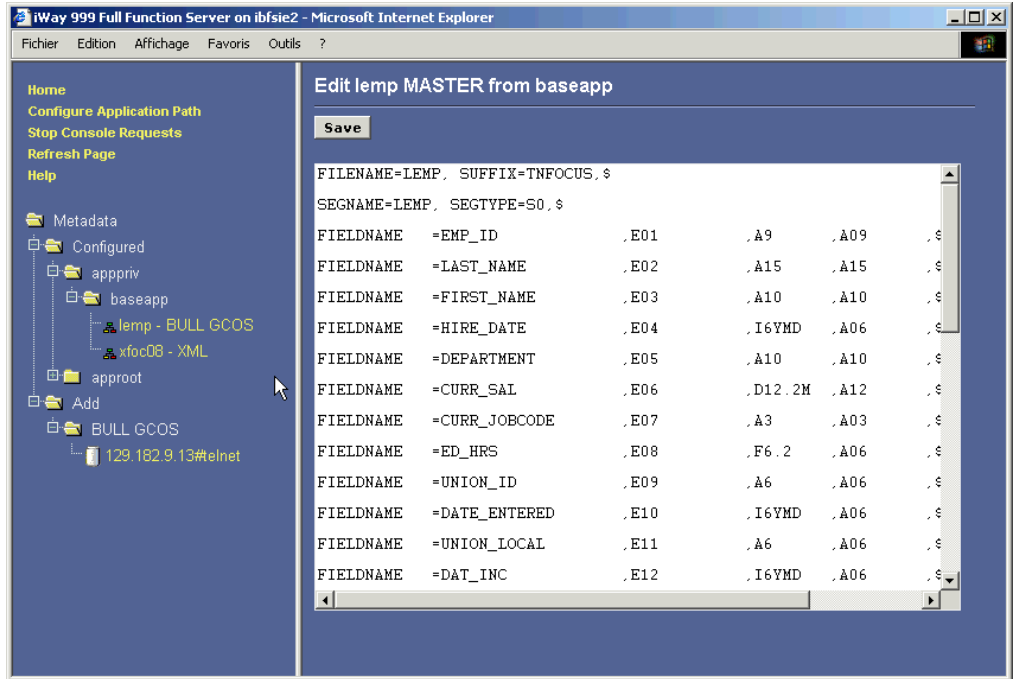
- Provide the valid values for Bull Server Name and Datasource as shown above, and click *Submit* to open the following window:



- Select *Application Directory* and *Synonym Name*, and click *Create Synonym*. After processing, the created synonym is stored in the given Application Directory:



4. Open the given Application Directory and click on the created synonym name to open the options window as shown above.
5. Select *Edit Master File*.



You can also edit the Access file by selecting *Edit Access File* from the options menu.

Syntax

How to Create a Synonym Manually

```
CREATE SYNONYM appname/synonym FOR datasource DBMS TNFOCUS [AT tnsname]
END
```

where:

appname

Is the 1 to 64 character application name space where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, then this application name must not be used.

synonym

Is an alias for the data source (maximum 64 characters for UNIX and Windows NT server platforms).

datasource

Is the fully qualified iWay for Bull RTE Master File name for the physical data structure residing on the Bull Server.

AT tnsname

Is the optional host name (or IP address) of the Bull Server used in the ENGINE ET CONNECTION_ATTRIBUTES command. It is also allowed to clone existing iWay for Bull RTE Master Files into the local server repository. In this case a fully qualified name for the iWay for Bull RTE Master File and the FTP Server access right must be given following the syntax:

```
AT server/ftp_user,ftp_password:'full path name of the iWay for Bull RTE Master File'
```

For example:

```
AT dps8/focus,ibfocus:'focus/qa/fso/wkslusers/focus/car.mas'  
END
```

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

CREATE SYNONYM creates a Master File and Access File which represents the server metadata.

Syntax**How to Use DROP SYNONYM**

```
DROP SYNONYM synonym
```

```
END
```

where:

```
synonym
```

Is the previously created synonym for the Bull GCOS data source.

```
END
```

Indicates the end of the command, and is required on a separate line in the stored procedure.

Customizing the Bull GCOS Environment

The Data Adapter for Bull GCOS provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Establishing an FTP Connection to an Intermediate Server

You can use the SET FTPSERVER command to set parameters for use when establishing an FTP connection to an intermediate server that will receive files from the Data Server for Bull GCOS. This command can be stored in the user profile (for example, EDASPROF.PRF) or in a FOCEXEC procedure.

Syntax How to Establish a FTP Connection to an Intermediate Server

```
ENGINE TNFOCUS SET FTPSERVER host/user,pass
```

where:

TNFOCUS

Indicates the Bull GCOS data source.

host

Defines the FTP Server host name or IP address. The FTP service port number if requested must follow the host name or IP address. For example, if the FTP Server (192.168.4.21) is listening on port number 7121, we must set: 192.168.4.21 7121.

user,pass

Defines the FTP Server user ID and password.

Example Setting Parameters With SET FTPSERVER

```
ENGINE TNFOCUS SET FTPSERVER 192.168.4.250/xy,11xy
```

where:

192.168.4.250

Is the FTP Server's IP address.

xy

Is the user ID.

11xy

Is the password.

Example Setting Parameters With FTP Server Service Address 7121

```
ENGINE TNFOCUS SET FTPSERVER 192.168.4.245 7121/xy,11xy
```

where:

```
192.168.4.245
```

Is the FTP Server's IP address.

```
7121
```

Is the FTP Server's service port address.

```
xy
```

Is the FTP user ID.

```
11xy
```

Is FTP password.

Specifying a Timeout Limit

Setting TIMEOUT enables you to designate the adapter connection timeout.

Syntax How to Issue the Timeout Command

```
ENGINE TNFOCUS SET TIMEOUT xx
```

where:

```
xx
```

Is the number of seconds before a timeout occurs. Timeout values range from 60 seconds to 300 seconds. The default value is 60 seconds.

When this command is used, the following message displays:

```
(FOC43515) TNFOCUS CONNECT TIMEOUT VALUE SET: xx (sec)
```

Note: If the value set is out of range, TIMEOUT is set to the default value.

Setting the Working Directory

You can use the SET FOCPATH command to set the iWay for Bull GCOS RTE working directory. This command is useful when a long path name setting cannot be done through the SET CONNECTION_ATTRIBUTES command line (the length of the line will exceed 80 characters). The directory path description must be quote protected for GCOS8 and GCOS7/XTA systems.

Syntax **How to Issue the FOCPATH Command**

```
ENGINE TNFOCUS SET FOCPATH path
```

where:

path

Is the full path leading to the iWay for Bull GCOS RTE working directory (library).

Example **Using TNFOCUS SET FOCPATH**

```
ENGINE TNFOCUS SET FOCPATH `FOCUS/QA/FSO/WKSLUSER/ADMIN`
```

This directory will be used as the iWay for Bull GCOS RTE working directory until a new SET FOCPATH is executed. To reset a previous setting of the iWay for Bull GCOS RTE directory path, use the command:

```
ENGINE TNFOCUS SET FOCPATH ``
```

Note: The iWay for Bull GCOS RTE working directory (FOCSLIB) is used by the server to store the iWay for Bull GCOS RTE session files and the FOCPATH setting must be consistent with the iWay for Bull GCOS RTE FOCSLIB setting.

Bull GCOS XTA (GCOS7) support may be introduced by XTA setting as given below. When this setting is done the XTA FTP7 is used to provide FTP services.

```
ENGINE TNFOCUS SET FOCPATH 'FOCUS/SL@XTA'
```

Displaying Current Adapter Settings

You can use the ? SET command to display the current adapter settings.

```
ENGINE TNFOCUS ? SET
```

An example of the messages that appears follows:

```
(FOC43573) CURRENT TNFOCUS ENGINE SETTINGS ARE:  
(FOC43574) REMOTE SERVER HOSTNAME IS -: dps7  
(FOC43582) REMOTE SERVER SERVICE PORT IS -: 7323  
(FOC43575) REMOTE SERVER ACCESS LOGON IS -: PGMSID  
(FOC43576) CLIENT ACCESSING THE SERVER IS -: pgmsid  
(FOC43577) REMOTE SERVER ACCESS TIMEOUT IS -: Not set  
(FOC43579) REMOTE SERVER CODEPAGE (only MSP) IS -: Not set  
(FOC43580) REMOTE FTP SERVER (only GCOS8) IS -: Not set  
(FOC43581) REMOTE SERVER WORKING DIRECTORY IS -: FOCUS.SD.SL
```

Enabling iWay for Bull Run-Time Environment (RTE)

You can use iWay for Bull RTE services from the enabled iWay front-end to interact with an iWay Server. The REMOTE EXEC facility is used for execution of iWay for Bull RTE stored procedures. The REMOTE PUT/GET facility is used to exchange files (upload/download) between the iWay Server and the iWay for Bull RTE from the enabled iWay Client front-end.

The following syntax executes the iWay for Bull RTE stored procedure 'proc'. The file 'proc' must be stored in the given FOCPATH parameter on Bull GCOS.

```
ENGINE TNFOCUS REMOTE EX proc
```

The following syntax ships and stores the client request 'proc.fex' into the iWay for Bull RTE execution environment. 'path' indicates the location from which the command file 'proc.fex' is read.

```
ENGINE TNFOCUS REMOTE PUT path/proc.fex
```

The following syntax downloads the file 'hold.ftm'. 'path' indicates the location where the 'hold.ftm' is stored.

```
ENGINE TNFOCUS REMOTE GET path/hold.ftm
```

The developer can mix REMOTE PUT/EX/GET to upload an iWay for Bull RTE request, submit it for execution to iWay for Bull RTE, and download the output on the client.

Note: Command files passed to iWay for Bull RTE must comply with FOCUS 6.7.1 (see the *iWay for Bull RTE* manual), and may not expect any input parameter or value to be supplied when executing.

CHAPTER 5

Getting Started in Informix C-ISAM, Micro Focus C-ISAM, VSAM, and Flat Files

Topics:

- Preparing the C-ISAM Environment
- Specifying the Location of C-ISAM Files

The Data Adapter for C-ISAM allows applications to access C-ISAM data sources. The adapter converts application requests into native C-ISAM statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

Preparing the C-ISAM Environment

The Informix Data Adapter for C-ISAM, Data Adapter for VSAM, and Data Adapter for Flat Files do not require setting any environment variables.

The Micro Focus Data Adapter for C-ISAM requires that a \$COBDIR UNIX environment variable, which identifies the installation directory to the operating system, be set. The variable must also be exported.

Syntax How to Identify the Micro Focus C-ISAM Installation Directory

```
COBDIR=path  
export COBDIR
```

where:

path

Is the location of the Micro Focus C-ISAM installation directory.

Specifying the Location of C-ISAM Files

Configure the adapter using the Web Console Adapter Add screen and click *Configure*.

You must specify the location of the data files you want to access in a supported server profile.

Syntax How to Specify the Location of Informix C-ISAM Files

```
FILEDEF CSAM0010 DISK filename.dat
```

where:

filename

Is the full path and file name of the data file.

Syntax How to Specify the Location of Micro Focus C-ISAM Files

```
FILEDEF CSAM0010 DISK filename
```

where:

filename

Is the full path and file name of the data file.

Note:

- For each data file there must be a corresponding index file, except for Micro Focus compression 8 files, which do not have a separate index file. The index is instead embedded in the data file.
- Read/write permissions must be given for both types of files.
- For the Informix Data Adapters for C-ISAM and the Micro Focus Data Adapters for C-ISAM, read-only access is supported.

Syntax How to Specify the Location of Flat Files on UNIX and Windows NT

FIXED data sets may be allocated with the FILEDEF command

```
FILEDEF FILE1 DISK filename [(LRECL lrecl RECFM recfm)]
```

where:

filename

Is the full path and file name of the data file.

lrecl

Is the record length in bytes. This parameter is optional. If you omit it, it is calculated based on metadata description.

The left parenthesis preceding the optional parameters is required.

recfm

Is the record format. Specify F for fixed format, V for variable format. This parameter is optional. If you omit it, the default is fixed format.

Syntax **How to Specify the Location of Flat Files on OS/390**

FIXED data sets may be allocated using JCL

```
//FILE1 DD DISP=SHR, DSN=dataset_name  
or file definition  
FILEDEF FILE1 DISK //'dataset_name' [(LRECL lrecl RECFM recfm)]
```

where:

dataset_name

Is the fully qualified dataset name.

Syntax **How to Specify the Location of Flat Files Residing on an HFS File System**

FIXED data sets may be allocated using file definition

```
FILEDEF FILE1 DISK filename [(LRECL lrecl RECFM recfm)]
```

where:

filename

Is the full path and file name of the data file.

lrecl

Is the record length in bytes. This parameter is optional. If you omit it, it is calculated based on metadata description.

The left parenthesis preceding the optional parameters is required.

recfm

Is the record format. Specify F for fixed format, V for variable format. This parameter is optional. If you omit it, the default is fixed format.

Note: The Data Adapter for Flat Files supports read/write access (write access can only be granted by INSERT only).

Syntax **How to Specify the Location of VSAM Files**

VSAM data sets may be allocated with JCL

```
//QAVSM DD DISP=SHR, DSN=cluster_name  
or with DYNAM  
DYNAM ALLOC FILE QAVSM DA cluster_name SHR REUSE
```

where:

cluster_name

Is the name of the VSAM cluster.

In order to use the alternate index, the path clusters must also be defined using DYNAM or JCL:

```
//VSAMFL1 DD DISP=OLD,DSN=EDABXV.VSAM.AXINDEX1.PATH  
//VSAMFL2 DD DISP=OLD,DSN=EDABXV.VSAM.AXINDEX2.PATH
```

where:

DD name

Is an alias in the Master File.

Note: The Data Adapter for VSAM supports full read/write access.

Specifying the Location of C-ISAM Files

CHAPTER 6

Getting Started in DATACOM

Topics:

- Preparing the DATACOM Environment
- Configuring the Data Adapter for DATACOM
- DATACOM Overview and Mapping Considerations
- Managing DATACOM Metadata
- Describing Multi-file Structures for DATACOM
- DATACOM Data Dictionary Master and Access Files
- Data Retrieval Logic for DATACOM

The Data Adapter for CA-DATACOM/DB for the OS/390 and z/OS platform allows applications to access DATACOM data sources. The adapter converts application requests into native DATACOM statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

Note: Throughout this manual, CA-DATACOM/DB is referred to as DATACOM.

Preparing the DATACOM Environment

The Data Adapter for CA-DATACOM/DB operates in the OS/390 and z/OS environment and issues standard DATACOM calls for record retrieval. The adapter uses DATACOM's Boolean Selection capabilities, enabling it to retrieve only records which satisfy a request. This reduces the number of I/Os involved in data retrieval.

Prior to configuring the adapter using the Web Console, you must allocate the CA-DATACOM load library data set to DDNAME STEPLIB of the server's ISTART JCL.

Configuring the Data Adapter for DATACOM

Using the Web Console, select the *Adapter Add* option from the main screen and select *Datacom* from the list provided under Add and then click *Configure*.

Declaring Connection Attributes

Users accessing the DATACOM DBMS are authenticated by the Operating System security package as well as any authentication performed by the DBMS itself.

User Requirements Table

First-level access to DATACOM data sources and fields is controlled by the DATACOM User Requirements Table (URT). DATACOM uses the URT to provide file and database-level security, and password protection for element-level security.

The URT used for your server structure must include all the DATACOM database IDs contained in the Access File. At run time, the data adapter will dynamically load the URT's load module.

Element-level Security

The Data Adapter for DATACOM supports element-level security by allowing specification of DATACOM passwords in the Access File. If an element is defined in the DATACOM control file with a security code, you must specify the two-character hex value of the code in the Access File.

DATACOM Overview and Mapping Considerations

DATACOM is a relational (or "flat file") database management system. The DATACOM DATA DICTIONARY stores and manages descriptive data about a data source.

To access DATACOM data sources, you describe them to server, using server terminology and attributes. The descriptions are kept in a Master File and an associated Access File.

The Master File tells the server how to interpret the DATACOM data structure. It identifies the DATACOM:

- Elements
- Fields (for each of those elements)
- Field type and field length (as DATACOM stores them)

The Access File creates a bridge between the server and DATACOM. It translates server terminology into DATACOM syntax by identifying the appropriate DATACOM:

- Database IDs
- Logical files
- Native key fields
- Element security information

When the server receives a request, it:

- Looks for the Master File describing that data source.
- Finds the suffix DATACOM in the Master File. This tells the server the requested data is in a DATACOM data source.
- Loads the Data Adapter for DATACOM module. The data adapter module then opens the Access File, in which it finds the DATACOM User Requirements Table (URT) name, the database ID(s), and file name(s).

The Data Adapter for DATACOM opens the URT and builds calls to DATACOM. The adapter retrieves the data and passes it to the server. The server then creates the answer set.

For details on describing data sources to a server and on Master Files and Access Files, and examples of DATACOM logical files, fields and elements, see *Mapping Structures in the Server* on page 6-5.

Data Structures

A DATACOM structure is a collection of one or more logical data sources, identified by a unique 3-byte ID number. Each logical data source contains data records with a defined set of fields, elements, and keys, and is identified by a unique, alphanumeric 3-byte table name.

A data record consists of one or more elements. An element is the smallest logical unit of data a program can request; it has a unique 5-byte name. Elements consist of one or more contiguous fields and may have an associated security code. Because a field may belong to more than one element, the elements themselves may overlap.

DATACOM locates records through keys. Each data source may have up to 999 keys (not necessarily contiguous) consisting of 1-180 fields which total a length of 180 bytes. Each key is identified by a 5-byte name and a 3-byte numeric ID. The use of keys in DATACOM's Compound Boolean Selection Facility is transparent to the user.

The table below is the structure of the sample PERSON data source. It contains two keys, three elements with fields that overlap, and seven unique fields.

EMPLOYEE NUMBER	EMPLOYEE NAME	STREET ADDRESS	CITY ADDRESS	STATE ADDRESS	ZIP CODE	SOCIAL SECURITY NUMBER
ELEMENT #1						
	ELEMENT #2 IDEMP					
			ELEMENT #3 ADEMP			
						ELEMENT #4 EMDTA

Note: The DATACOM DATA DICTIONARY listing of the PERSON data source does not identify the specific fields in each element. For this information, refer to the DATA DICTIONARY ELEMENT FIELD REPORT.

The Employee Record Element (EMDTA) contains all seven fields in the data source.

The Employee Address Element (ADEMP) contains six fields that represent the employee ID and address. These are:

[NUMBER](#)
[NAME](#)
[STREET_ADDRESS](#)
[CITY_ADDRESS](#)
[STATE_ADDRESS](#)
[ZIP_CODE_LOC](#)

The Employee Identification Element (IDEMP) contains the NUMBER and NAME fields. All fields within each element are contiguous.

Mapping Structures in the Server

DATACOM terms equate to server terms as follows:

DATACOM Master File Definition	Server Master File Attributes
File	SEGMENT
Element	ALIAS for groups of fields
Field	FIELD
Field Length (LNGTH)	ACTUAL (DATACOM Format) USAGE (Server Format)

When you describe a DATACOM data source in a Master File:

- You can choose one or more elements from a DATACOM data source to build a single server segment. You do not have to specify all the elements in a particular data source.
- You must define each field in the element in the order in which it appears in the DATACOM data source. A DATACOM field becomes a server field.
- You can define two or more DATACOM elements that contain overlapping fields. However, you must define each overlapping field to the server with a unique name each time it appears in an element.
- Each DATACOM field you define to the server must have the 5-byte element name to which the DATACOM field belongs, together with a unique qualifier.
- For each DATACOM field, the DATACOM field length becomes the ACTUAL field length.
- The number of decimal positions that DATACOM indicates becomes the number of decimal positions in the USAGE format.

Example Using the DATA DICTIONARY Element Field Report for EMDTA

The following example represents the DATA DICTIONARY Element Field Report for the Employee Record Element (EMDTA) from the PERSON data source. The table in *Element Field Report Headings* on page 6-7 explains the relevant column headings in the report.

```
ENTITY-TYPE: ELEMENT NAME: PERSONNEL.EMPLOYEE (001) PROD DESC: EMPLOYEE RE
AUTHOR: JOHN DOE CONTROLLER: HR DESIGNER COPY-VERSION:
FILE-NAME: PERSON-MASTER DBNAME: PMF ID: 002 DESC: PERSONNEL MASTER FILE

LV C FIELD-NAME..PAREN-NAME.DISPL LNGTH T J S DEC RPFAC
DESCRIPTION...VALUE
ALC-NAME COMPILER-NAME.....LANGUAGE-COMMENT
```

DATAKOM Overview and Mapping Considerations

```
01 S NUMBER 0 5 N R N 00001 EMPLOYEE NUMBER  
EMNUMBER EM-IDENTIFICATION-NUMBER  
01 S NAME 5 24 C L N 00001 EMPLOYEE NAME  
EMNAME EM-IDENTIFICATION-NAME  
01 S STREET-ADDRESS 29 24 C L N 00001 EMPLOYEE STREET ADDRESS  
EMADDR EM-STREET-ADDRESS  
01 S CITY-ADDRESS 53 15 C L N 00001 EMPLOYEE CITY  
EMCITY EM-CITY-ADDRESS  
01 S STATE-ADDRESS 68 2 C L N 00001 EMPLOYEE STATE CODE  
EMSTATE EM-STATE-ADDRESS  
01 S ZIP-CODE-LOC 70 5 C L N 00001 EMPLOYEE ZIP CODE  
EMZIP EM-ZIP-CODE-LOC  
01 S SOCIAL-SECURITY 75 5 D R Y 00001 EMPLOYEE SOCIAL SECURITY  
EMSSN EM-SOCIAL-SECURITY-NUMBER
```


Reference Element Field Report Headings

Column Heading	Definition
LV	COBOL copybook level corresponding to the field.
C	Field class: S= Simple C= Compound V= Value F=Filler
FIELD-NAME	DATACOM field name.
PAREN-NAME	The PARENT field to which the field belongs if the field class is C or V, or if this is a redefined field.
DISPL	Byte displacement of the field in the data source.
LNTH	Field length in bytes.
T	Field Type: B=Binary C=Character D=Packed decimal H=2-character hexadecimal field N=Zoned decimal T=PL/1 bit representation
J	Justification: L=Left R=Right
S	Signed field: Y=Yes N=No
DEC	Number of decimal places for a numeric field.
RPFAC	Number of times the field can repeat.
DESCRIPTION	Description of the field.
VALUE	Value of a COBOL copybook VALUE CLAUSE when the Field Class is V.

The following Master File and Access File that describe the Employee Record Element from the DATACOM PERSON data source.

Master File

```
FILENAME=PERSON , SUFFIX=DATACOM
SEGNAME=PERSON , SEGTYPE=S , $
FIELD=EMP_NO , ALIAS=EMDTA.EN , USAGE=P6 , ACTUAL=Z5 , $
FIELD=NAME , ALIAS=EMDTA.NAME , USAGE=A24 , ACTUAL=A24 , $
FIELD=STREET , ALIAS=EMDTA.STR , USAGE=A24 , ACTUAL=A24 , $
FIELD=CITY , ALIAS=EMDTA.CITY , USAGE=A15 , ACTUAL=A15 , $
FIELD=STATE , ALIAS=EMDTA.STAT , USAGE=A2 , ACTUAL=A2 , $
FIELD=ZIP , ALIAS=EMDTA.ZIP , USAGE=A5 , ACTUAL=A5 , $
FIELD=SSNO , ALIAS=EMDTA.SSNO , USAGE=P9L , ACTUAL=P5 , $
```

Access File

```
TRACE=NO , USERTABLE=URTLOAD , $
SEGNAME=PERSON , DBID=002 , TABLENAME=PMF , KEYNAME=EMPNO , $
```

Managing DATACOM Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a Master File and Access File that describe the structure of the data source and the server mapping of the DATACOM data types. The logical description of a DATACOM file is stored in a Master File, which describes the field layout. The physical attributes of the DATACOM file, such as the database number, file number, descriptors, and security are stored in the Access File.

Master Files for DATACOM

There are three types of Master File declarations.

Each declaration must begin on a separate line. A declaration consists of attribute-value pairs separated by commas. A declaration can span as many lines as necessary, as long as no single keyword-value pair spans two lines.

Do not use system or reserved words as names for files, segments, fields, or aliases. Specifying a reserved word generates syntax errors.

File Attributes

Each Master File begins with a file declaration. The file declaration has two attributes:

`FILENAME (FILE)`

Identifies the Master File.

`SUFFIX`

Identifies the data adapter needed to interpret the request.

The syntax is

```
FILE[NAME]=filename, SUFFIX=DATACOM [, $]
```

where:

filename

Is the file name for the Master File. The file name can consist of a maximum of eight alphanumeric characters. The file name should start with a letter and be representative of the table or view contents.

`DATACOM`

Is the value for the Data Adapter for CA-DATACOM/DB.

Segment Attributes

Each table described in a Master File requires a segment declaration. The segment declaration consists of at least two attributes:

SEGNAME

Identifies one table.

SEGTYPE

Identifies the type of segment sequencing.

The syntax for a segment declaration is

```
SEGNAME=segname , SEGTYPE=type [ , $ ]
```

where:

segname

Is the segment name that serves as a link to the actual DATACOM table name. It can consist of a maximum of 8 alphanumeric characters. It may be the same as the name chosen for FILENAME, the actual table name, or an arbitrary name.

The SEGNAME value in the Master File must be identical to the SEGNAME value specified in the Access File.

type

Indicates the type of segment sequencing:

S indicates the segments are logically sequenced in low to high order.

U indicates a unique segment.

Field Attributes

Each row in a table may consist of one or more columns. These columns are described in the Master File as fields with the following primary field attributes:

FIELDNAME

Identifies the name of a field.

ALIAS

Identifies the DATACOM 5-byte element short name to which the field belongs, along with a unique qualifier.

USAGE

Identifies how to display a field on reports.

ACTUAL

Identifies the datatype and length in bytes for a field.

You can get values for these attributes from the DATACOM DATA DICTIONARY Element Field Report.

The syntax for a field declaration is:

```
FIELD[NAME]=fieldname, ALIAS=elementname.qualifier,  
[USAGE=]display_format, [ACTUAL=]storage_format , $
```

where:

fieldname

Is the unqualified name of the field. You must describe the fields in the order in which they appear in the DATACOM element. You can find the DATACOM field names, their relative position within the element, and the DATACOM formats in the DATA DICTIONARY Element Field Report. This value must be unique within the Master File. The name can consist of a maximum of 48 alphanumeric characters (including any file name and segment name qualifiers and qualification characters you may later prefix to them in your requests). The name must begin with a letter. Special characters and embedded blanks are not recommended.

Note that some elements contain overlapping fields. You must define overlapping fields for those elements in which they appear, and assign a unique *fieldname* each time.

It is not necessary to describe all the columns of the DATACOM table in your Master File.

elementname

Is the DATACOM 5-byte element short name, followed by a period, to which the field belongs. You can find the name in the DATACOM DATA DICTIONARY listing of Master Files.

qualifier

Is the qualifier that is separated from the element name by a period. The qualifier is used to make the ALIAS unique and provide the field with an alternate identification. The total length of the ALIAS, including the element name and qualifier, cannot exceed 12 characters.

display_format

Is the display format. The value must include the field type and length and may contain edit options.

The data type of the display format must be identical to that of the ACTUAL format. For example, a field with an alphanumeric USAGE data type must have an alphanumeric ACTUAL data type.

Fields or columns with decimal or floating point data types must be described with the correct scale (s) and precision (p). Scale is the number of positions to the right of the decimal point. Precision is the total length of the field.

For the server, the total display length of the field or column *includes* the decimal point and negative sign. In SQL, the total length of the field or column *excludes* the decimal point and negative sign.

For example, a column defined as DECIMAL(5,2) would have a USAGE attribute of P7.2 to allow for the decimal point and a possible negative sign.

storage_format

Is the storage format of the DATACOM data type and length, in bytes, for the field. You can find this under LENGTH in the field report.

Access Files for DATACOM

Each Master File must have a corresponding Access File. The file name of the Access File must be the same as that used for the Master File.

The Access File serves as a link between the server and DATACOM. It stores such information as the URT name, DATACOM Database ID(s), DATACOM table name(s), keyname(s), and element security information. In addition, the Access File provides you with the TRACE parameter, a useful debugging tool.

The Access File contains three types of logical records:

- Header record
- Segment record
- Element record

Each consists of a list of keyword and value pairs, separated by commas and terminated by a comma and dollar sign (,\$). The list is free form and may span several lines; keywords may be in any order.

Syntax **How to Use the Header Logical Record**

The header record contains the TRACE and USERTABLE attributes. The syntax is:

```
TRACE=NO|YES, USERTABLE=name, $
```

where:

YES

Indicates a complete trace of executed DATACOM commands. It is recommended that you use TRACE only for debugging purposes.

NO

Indicates no trace. This value is the default.

name

Is the name of the load module that supplies the DATACOM User Requirements Table (URT) information to be dynamically loaded. The URT must contain all the database IDs and file specified in the Master File, or you will not have access to the records.

Syntax **How to Use the Segment Logical Record**

The segment record contains these attributes:

```
SEGNAME=segname, DBID=dbid, TABLENAME=tablename, [KEYFLD=keyfld,  
IXFLD=ixfld,] KEYNAME=keyname, $
```

where:

segname

Identifies the segment name. The Master File SEGMENT attribute is the same as the SEGNAME attribute in the Access File.

dbid

Identifies the numeric, 3-byte DATACOM database ID.

tablename

Identifies the 3-byte DATACOM table name to which the segment identified by SEGNAME corresponds. You can find this name in the DATA DICTIONARY Element Field Report, under DBNAME.

keyfld

Is the server field name from the parent segment of a cross-reference; it is mandatory for everything but the root segment. You may also specify multiple field names, concatenated with the forward slash symbol (/) without blanks. The keyword value must be contiguous, and on the same line. The key list can span multiple lines.

ixfld

Is the server field name in the cross-referenced segment that establishes the cross-reference. It is mandatory for all but the root segment. The value(s) of these field(s) must have USAGE and ACTUAL formats comparable to the KEYFLD.

You may also specify multiple field names, concatenated with the forward slash symbol (/) without blanks. The keyword value may not span more than one line. The key list can span multiple lines.

keyname

Is the native key. This is in the DATA DICTIONARY Indented Report. In that report, MN stands for mandatory native.

Note: KEYNAME does not have to be the native key. If you wish to use a different keyname, that is acceptable. However, be aware that you may not get all the records.

Syntax

How to Specify the Element Logical Record Security

You must specify an element logical record if an element is defined to DATACOM with a security code. The syntax is:

```
ELEMENT=element, SECURITY=security, $
```

where:

element

The five-byte name of the element.

security

The two-position hexadecimal value of the DATACOM 1-byte security code.

Describing Multi-file Structures for DATACOM

You can describe many different DATACOM data sources in one Master File and Access File pair. However, in the data adapter, there must be at least one field in common between any parent and descendant pair of data sources. Each set of related fields must also have comparable USAGE and ACTUAL formats.

Each time you add a descendant segment, you must specify the PARENT attribute in the segment record. This will identify hierarchical relationships between the data sources.

In addition, you must specify the relationship between the fields in the Access File with the KEYFLD and IXFLD attributes:

- The value of the KEYFLD can be a simple field, a DEFINE field, or a list of fields.
- The value of IXFLD can be a simple field, or a list of fields.
- The length, and the format, of KEYFLD and IXFLD must be the same.

There are advantages to defining multiple data sources in a single structure:

- The multi-file structure creates a view of the DATACOM data source.
- You can describe up to 64 separate but related DATACOM logical files as segments in a single Master File. This will allow you to issue a request from any or all of the 64 segments defined in a single Master File without issuing a JOIN command.

To illustrate this concept, we will use the DATACOM data sources PERSON and PAYROLL.

The PAYROLL data source contains information about the wages, commissions, and taxes for each person in the PERSON data source.

Master File

```
FILENAME=PAYROLL , SUFFIX=DATACOM
SEGNAME=PAYROLL , SEGTYPE=S , $
FIELD=ENUM      , ALIAS=PYIDT . EN      , USAGE=P6      , ACTUAL=Z5      , $
FIELD=ACTCODE   , ALIAS=PYIDT . ACODE   , USAGE=A1      , ACTUAL=A1      , $
FIELD=ACTSTATUS , ALIAS=PYIDT . ASTATU , USAGE=A1      , ACTUAL=A1      , $
FIELD=CURRATE   , ALIAS=FIGSS . CRATE   , USAGE=P10 . 2 , ACTUAL=Z8      , $
FIELD=YTDWAGES  , ALIAS=FIGSS . YWAGES  , USAGE=P10 . 2 , ACTUAL=Z8      , $
FIELD=YTDCOMM   , ALIAS=FIGSS . YCOMM   , USAGE=P10 . 2 , ACTUAL=Z8      , $
FIELD=YTDTAXES  , ALIAS=FIGSS . YTAXES  , USAGE=P10 . 2 , ACTUAL=Z8      , $
```

Access File

```
TRACE=NO , USERTABLE=URTLOAD , $
SEGNAME=PAYROLL , DBID=002 , TABLENAME=PAY , KEYNAME=EMPNO , $
```

Multi-file Master File

Consider an application that contains both the PERSON and PAYROLL data sources. It is reasonable to generate reports that identify the current salary rate for all employees, or a list of all the employees who have a particular salary rate.

For demonstration purposes, we will define both these DATACOM logical files in one Master File. In all but the top (or root) segment, you must specify the PARENT attribute. PARENT establishes the relationship between two segments. The syntax is:

```
PARENT=segname
```

where:

```
segname
```

The name of the child segment's parent. If you do not specify a PARENT attribute, it will default to the child segment's immediate predecessor as the parent.

The following example shows the new multi-file Master File with the PARENT attribute added to the descendent segment. The filename PERSPAY identifies the entire Master File.

```
FILENAME=PERSPAY, SUFFIX=DATACOM
SEGNAME=PERSON, SEGTYPE=S, $
FIELD=EMP_NO, ALIAS=EMDTA.EN, USAGE=P6, ACTUAL=Z5, $
FIELD=NAME, ALIAS=EMDTA.NAME, USAGE=A24, ACTUAL=A24, $
FIELD=STREET, ALIAS=EMDTA.STR, USAGE=A24, ACTUAL=A24, $
FIELD=CITY, ALIAS=EMDTA.CITY, USAGE=A15, ACTUAL=A15, $
FIELD=STATE, ALIAS=EMDTA.STAT, USAGE=A2, ACTUAL=A2, $
FIELD=ZIP, ALIAS=EMDTA.ZIP, USAGE=A5, ACTUAL=A5, $
FIELD=SSNO, ALIAS=EMDTA.SSNO, USAGE=P9L, ACTUAL=P5, $

SEGNAME=PAYROLL, SEGTYPE=S, PARENT=PERSON, $
FIELD=ENUM, ALIAS=PYIDT.EN, USAGE=P6, ACTUAL=Z5, $
FIELD=ACTCODE, ALIAS=PYIDT.ACODE, USAGE=A1, ACTUAL=A1, $
FIELD=ACTSTATUS, ALIAS=PYIDT.ASTATU, USAGE=A1, ACTUAL=A1, $
FIELD=CURRATE, ALIAS=FIGSS.CRATE, USAGE=P10.2, ACTUAL=Z8, $
FIELD=YTDWAGES, ALIAS=FIGSS.YWAGES, USAGE=P10.2, ACTUAL=Z8, $
FIELD=YTDCOMM, ALIAS=FIGSS.YCOMM, USAGE=P10.2, ACTUAL=Z8, $
FIELD=YTDTAXES, ALIAS=FIGSS.YTAXES, USAGE=P10.2, ACTUAL=Z8, $
```

Multi-file Access File

The Access File for a multi-file structure must have a segment record for each logical segment you define in the Master File.

Each segment record, other than the root, must contain the KEYFLD and IXFLD attributes in addition to the Access File attributes described earlier. These new attributes identify the field(s) in an embedded cross-reference. The common field(s) serve as the cross-referenced link between the two data sources.

The PAYROLL data source is the child, or cross-referenced, segment in the following example. This is important because you identify cross-referencing fields by specifying KEYFLD and IXFLD in the cross-referenced segment record.

KEYFLD

The field name in the parent or host segment, or a list of up to five fields separated by a forward slash symbol (/).

IXFLD

The field name in the child or cross-referenced segment, or a list of up to five fields separated by a forward slash symbol (/).

In the example, the field EMP_NO in the PERSON data source is related to the field ENUM in the PAYROLL data source. Both have the same USAGE and ACTUAL data format and length (P6 and Z5); this is a server requirement.

Add the segment record for the PAYROLL data source to the Access File, including the KEYFLD and IXFLD attributes:

```
TRACE=NO,USERTABLE=URTLLOAD,$
SEGNAME=PERSON,DBID=002,tablename=PMF,KEYNAME=EMPNO,$

SEGNAME=PAYROLL,DBID=002,tablename=PAY,KEYNAME=EMPNO,

KEYFLD=EMP_NO,IXFLD=ENUM,$
```

Using Multiple Fields to Cross-reference Data Sources

With the Data Adapter for DATACOM, you can use up to five fields to establish a relationship or cross-reference between data sources.

For example, matching permutations of values of Z/Y/X/W/V in the KEYFLD and IXFLD attributes can be used to create new cross-referenced fields in the Access File. This is sometimes referred to as using concatenated keys.

To illustrate how to use concatenated keys, we will alter the PERSPAY multi-file Master and Access Files as shown in the following example:

1. Replace the field EMP_NO in the PERSON segment with the LAST_NAME and FIRST_NAME fields.
2. In the PAYROLL segment, replace the field ENUM with LN and FN.
3. Separate each field that is part of the cross-reference with a forward slash symbol (/).

```
FILENAME=PERPAY,SUFFIX=DATACOM
SEGNAME=PERSON,SEGTYPE=S,$
FIELD=LAST_NAME,ALIAS=EMDTA.LN,USAGE=A15,ACTUAL=A15,$
FIELD=FIRST_NAME,ALIAS=EMDTA.FN,USAGE=A10,ACTUAL=A10,$
FIELD=NAME,ALIAS=EMDTA.NAME,USAGE=A24,ACTUAL=A24,$
FIELD=STREET,ALIAS=EMDTA.STR,USAGE=A24,ACTUAL=A24,$
FIELD=CITY,ALIAS=EMDTA.CITY,USAGE=A15,ACTUAL=A15,$
FIELD=STATE,ALIAS=EMDTA.STAT,USAGE=A2,ACTUAL=A2,$
FIELD=ZIP,ALIAS=EMDTA.ZIP,USAGE=A5,ACTUAL=A5,$
FIELD=SSNO,ALIAS=EMDTA.SSNO,USAGE=P9L,ACTUAL=P5,$

SEGNAME=PAYROLL,SEGTYPE=S,PARENT=PERSON,$
FIELD=LN,ALIAS=PYIDT.LNAME,USAGE=A15,ACTUAL=A15,$
FIELD=FN,ALIAS=PYIDT.FNAME,USAGE=A10,ACTUAL=A10,$
FIELD=ACTCODE,ALIAS=PYIDT.ACODE,USAGE=A1,ACTUAL=A1,$
FIELD=ACTSTATUS,ALIAS=PYIDT.ASTATU,USAGE=A1,ACTUAL=A1,$
FIELD=CURRATE,ALIAS=FIGSS.CRATE,USAGE=P10.2,ACTUAL=Z8,$
FIELD=YTDWAGES,ALIAS=FIGSS.YWAGES,USAGE=P10.2,ACTUAL=Z8,$
FIELD=YTDCOMM,ALIAS=FIGSS.YCOMM,USAGE=P10.2,ACTUAL=Z8,$
FIELD=YTDTAXES,ALIAS=FIGSS.YTAXES,USAGE=P10.2,ACTUAL=Z8,$
```

These multiple fields will now become the KEYFLD and IXFLD attributes in the Access File.

Separate each field from the next with the forward slash symbol (/), and add these attributes to the PAYROLL segment record in the Access File:

```
TRACE=NO,USERTABLE=URTLOAD,$
SEGNAME=PERSON,DBID=002,TABLENAME=PMF,KEYNAME=LNAME,$
SEGNAME=PAYROLL,DBID=002,TABLENAME=PAY,KEYNAME=LNAME,
KEYFLD=LAST_NAME/FIRST_NAME,IXFLD=LN/FN,$
```

DATAKOM Data Dictionary Master and Access Files

You can describe many different DATAKOM data sources in one Master File and Access File pair. However, in the data adapter, there must be at least one field in common between any parent and descendant pair of data sources. Each set of related fields must also have comparable USAGE and ACTUAL formats.

The following are Master Files and Access Files, and examples of DATAKOM logical files, fields, and elements from the DATAKOM Data Dictionary Database listing.

- Data Dictionary PERSON-MASTER Indented Report
- Data Dictionary Element Field Report for EMDTA
- PERSON Master File and Access File
- Data Dictionary PAYROLL-MASTER Indented Report
- Data Dictionary Element Field Report for PAYROLL
- PAYROLL Master File and Access File
- PERSPAY Master File and Access File

Data Dictionary PERSON-MASTER Indented Report

```
AREA PEOPLE 001 P PERSONNEL AREA PMF
FILE PERSON-MASTER 001 P PERSONNEL MASTER FILE PMF 002
RECORD PERSONNEL 001 P PERSONNEL RECORD
KEY PERSONNEL.NUMBER 001 P EMPLOYEE NUMBER KEY EMPNO 001 MN
KEY PERSONNEL.STATE-ZIP 001 P EMPLOYEE STATE ZIP KEY STZIP 002
ELEMENT PERSONNEL.EMPLOYEE 001 P EMPLOYEE RECORD ELEMENT EMDTA
ELEMENT PERSONNEL.FULL ADDRESS 001 P EMPLOYEE ADDRESS ELEMEN ADEMP
ELEMENT PERSONNEL.IDENTIFICATION 001 P EMPLOYEE IDENTIFICATION IDEMP
FIELD PERSONNEL.CITY-ADDRESS 001 P EMPLOYEE CITY
FIELD PERSONNEL.NAME 001 P EMPLOYEE NAME
FIELD PERSONNEL.NUMBER 001 P EMPLOYEE NUMBER
FIELD PERSONNEL.SOCIAL-SECURITY 001 P EMPLOYEE SOCIAL SECURITY NUMBER
FIELD PERSONNEL.STREET-ADDRESS 001 P EMPLOYEE STREET ADDRESS
FIELD PERSONNEL.ZIP-CODE-LOC 001 P EMPLOYEE ZIP CODE
```

Data Dictionary Element Field Report for EMDTA

```

ENTITY-TYPE:ELEMENT NAME:PERSONNEL.EMPLOYEE (001)PROD DESC:EMPLOYEE RE
AUTHOR:JOHN DOE CONTROLLER:HR DESIGNER COPY-VERSION:
FILE-NAME: PERSON-MASTER DBNAME: PMF ID: 002 DESC: PERSONNEL MASTER FILE
LV C FIELD-NAME..PAREN-NAME.DISPL LNTH T J S DEC RPFAC
DESCRIPTION.....VALUE
ALC-NAME COMPILER-NAME.....LANGUAGE-COMMENT
01 S NUMBER 0 5 N R N 00001 EMPLOYEE NUMBER
EMNUMBER EM-IDENTIFICATION-NUMBER
01 S NAME 5 24 C L N 00001 EMPLOYEE NAME
EMNAME EM-IDENTIFICATION-NAME
01 S STREET-ADDRESS 29 24 C L N 00001 EMPLOYEE STREET ADDRESS
EMADDR EM-STREET-ADDRESS
01 S CITY-ADDRESS 53 15 C L N 00001 EMPLOYEE CITY
EMCITY EM-CITY-ADDRESS
01 S STATE-ADDRESS 68 2 C L N 00001 EMPLOYEE STATE CODE
EMSTATE EM-STATE-ADDRESS
01 S ZIP-CODE-LOC 70 5 C L N 00001 EMPLOYEE ZIP CODE
EMZIP EM-ZIP-CODE-LOC
01 S SOCIAL-SECURITY 75 5 D R Y 00001 EMPLOYEE SOCIAL SECURITY
EMSSN EM-SOCIAL-SECURITY-NUMBER

```

PERSON Master File and Access File**Master File**

```

FILENAME=PERSON,SUFFIX=DATACOM
SEGNAME=PERSON,SEGTYPE=S,$
FIELD=EMP_NO ,ALIAS=EMDTA.EN ,USAGE=P6 ,ACTUAL=Z5 , $
FIELD=NAME ,ALIAS=EMDTA.NAME ,USAGE=A24 ,ACTUAL=A24 , $
FIELD=STREET ,ALIAS=EMDTA.STR ,USAGE=A24 ,ACTUAL=A24 , $
FIELD=CITY ,ALIAS=EMDTA.CITY ,USAGE=A15 ,ACTUAL=A15 , $
FIELD=STATE ,ALIAS=EMDTA.STAT ,USAGE=A2 ,ACTUAL=A2 , $
FIELD=ZIP ,ALIAS=EMDTA.ZIP ,USAGE=A5 ,ACTUAL=A5 , $
FIELD=SSNO ,ALIAS=EMDTA.SSNO ,USAGE=P9L ,ACTUAL=P5 , $

```

Access File

```

TRACE=NO,USERTABLE=URTLOAD,$
SEGNAME=PERSON,DBID=002,TABLENAME=PMF,KEYNAME=EMPNO,$

```

Data Dictionary PAYROLL-MASTER Indented Report

```
AREA PEOPLE 001 P PAYROLL AREA PAY
FILE PAYROLL-MASTER 001 P PAYROLL MASTER FILE PAY 001
RECORD PAYROLL 001 P PAYROLL RECORD

KEY PAYROLL.NUMBER 001 P EMPLOYEE NUMBER KEY EMPNO 001 MN
ELEMENT PAYROLL.ACTIVITY-CODE 001 P EMPLOYEE NUMBER ELEMENT PYIDT
ELEMENT PAYROLL.FIGURES 001 P EMPLOYEE PAY FIGURES FIGSS
ELEMENT PAYROLL.RECORD 001 P EMPLOYEE PAYROLL RECORD PAYRC
FIELD PAYROLL.ACTIVITY-CODE 001 P EMPLOYEE CODE
FIELD PAYROLL.ACTIVITY-STATUS 001 P EMPLOYEE PAY STATUS
FIELD PAYROLL.CURRENT-RATE 001 P EMPLOYEE RATE
FIELD PAYROLL.NUMBER 001 P EMPLOYEE NUMBER
FIELD PAYROLL.YTD-COMMISSION 001 P EMPLOYEE COMMISSION
FIELD PAYROLL.YTD-TAX 001 P EMPLOYEE YEAR T DATE TAXES
FIELD PAYROLL.YTD-WAGES 001 P EMPLOYEE YTD WAGES
```

Data Dictionary Element Field Report for PAYROLL

```
ENTITY-TYPE:ELEMENT NAME:PAYROLL.RECORD (001)PROD DESC:EMPLOYEE PAYROLL
RECORD ELEMEN
```

```
FILE-NAME: PAYROLL-MASTER DBNAME: PAY ID: 001 DESC: PAYROLL MASTER FILE
```

```
LV C FIELD-NAME..PARENT-NAME.DISPL LNTH T J S DEC RPFAC
DESCRIPTION...VALUE
ALC-NAME COMPILER-NAME.....LANGUAGE-COMMENT
```

```
01 S NUMBER 0 5 N R N 00001 EMPLOYEE NUMBER
RCNUM RC-NUMBER
01 S ACTIVITY-CODE 5 1 C L N 00001 EMPLOYEE CODE
RCCODE RC-ACTIVITY-CODE
01 S ACTIVITY-STATUS 6 1 C L N 00001 EMPLOYEE PAY STATUS
RCSTATUS RC-ACTIVITY-STATUS
01 S CURRENT-RATE 7 8 N R N 00001 EMPLOYEE RATE
RCRATE RC-CURRENT-RATE
01 S YTD-WAGES 15 8 N R N 00001 EMPLOYEE YTD WAGES
RCWAGES RC-YTD-WAGE
01 S YTD-COMMISSION 23 8 N R N 00001 EMPLOYEE COMMISSION
RCCOMM RC-YTD-COMMISSION
01 S YTD-TAX 31 8 N R N 00001 EMPLOYEE YEAR T DATE TAXES
RCTAX RC-YTD-TAX
```

PAYROLL Master File and Access File

Master File

```

FILENAME=PAYROLL , SUFFIX=DATACOM
SEGNAME=PAYROLL , SEGTYPE=S , $
FIELD=ENUM , ALIAS=PYIDT.EN , USAGE=P6 , ACTUAL=Z5 , $
FIELD=ACTCODE , ALIAS=PYIDT.ACODE , USAGE=A1 , ACTUAL=A1 , $
FIELD=ACTSTATUS , ALIAS=PYIDT.ASTATU , USAGE=A1 , ACTUAL=A1 , $
FIELD=CURRATE , ALIAS=FIGSS.CRATE , USAGE=P10.2 , ACTUAL=Z8 , $
FIELD=YTDWAGES , ALIAS=FIGSS.YWAGES , USAGE=P10.2 , ACTUAL=Z8 , $
FIELD=YTDCOMM , ALIAS=FIGSS.YCOMM , USAGE=P10.2 , ACTUAL=Z8 , $
FIELD=YTDTAXES , ALIAS=FIGSS.YTAXES , USAGE=P10.2 , ACTUAL=Z8 , $

```

Access File

```

TRACE=NO , USERTABLE=URTLLOAD , $
SEGNAME=PAYROLL , DBID=002 , TABLENAME=PAY , KEYNAME=EMPNO , $

```

PERSPAY Master File and Access File

Master File

```

FILENAME=PERSPAY , SUFFIX=DATACOM
SEGNAME=PERSON , SEGTYPE=S , $
FIELD=EMP_NO , ALIAS=EMDTA.EN , USAGE=P6 , ACTUAL=Z5 , $
FIELD=NAME , ALIAS=EMDTA.NAME , USAGE=A24 , ACTUAL=A24 , $
FIELD=STREET , ALIAS=EMDTA.STR , USAGE=A24 , ACTUAL=A24 , $
FIELD=CITY , ALIAS=EMDTA.CITY , USAGE=A15 , ACTUAL=A15 , $
FIELD=STATE , ALIAS=EMDTA.STAT , USAGE=A2 , ACTUAL=A2 , $
FIELD=ZIP , ALIAS=EMDTA.ZIP , USAGE=A5 , ACTUAL=A5 , $
FIELD=SSNO , ALIAS=EMDTA.SSNO , USAGE=P9L , ACTUAL=P5 , $
SEGNAME=PAYROLL , SEGTYPE=S , PARENT=PERSON , $
FIELD=ENUM , ALIAS=PYIDT.EN , USAGE=P6 , ACTUAL=Z5 , $
FIELD=ACTCODE , ALIAS=PYIDT.ACODE , USAGE=A1 , ACTUAL=A1 , $ }
FIELD=ACTSTATUS , ALIAS=PYIDT.ASTATU , USAGE=A1 , ACTUAL=A1 , $
FIELD=CURRATE , ALIAS=FIGSS.CRATE , USAGE=P10.2 , ACTUAL=Z8 , $
FIELD=YTDWAGES , ALIAS=FIGSS.YWAGES , USAGE=P10.2 , ACTUAL=Z8 , $
FIELD=YTDCOMM , ALIAS=FIGSS.YCOMM , USAGE=P10.2 , ACTUAL=Z8 , $
FIELD=YTDTAXES , ALIAS=FIGSS.YTAXES , USAGE=P10.2 , ACTUAL=Z8 , $

```

Access File

```

TRACE=NO , USERTABLE=URTLLOAD , $
SEGNAME=PERSON , DBID=002 , TABLENAME=PMF , KEYNAME=EMPNO , $
SEGNAME=PAYROLL , DBID=002 , TABLENAME=PAY , KEYNAME=EMPNO , $
KEYFLD=EMP_NO , IXFLD=ENUM , $

```

Data Retrieval Logic for DATACOM

This section describes the two types of DATACOM record retrieval commands generated by the Data Adapter for DATACOM:

- **Sequential Retrieval Commands: GSETL and GETIT.** These commands are issued by the data adapter only for the root of the accessed subtree, and only if the request did not specify any record selection criteria on that segment.
- **Compound Boolean Selection Commands: SELFR and SELNR.** These commands are issued for DATACOM records when there are available selection criteria.

GSETL and GETIT

When the data adapter determines sequential retrieval needs to be done for the root segment, it issues a GSETL command. It uses the KEYNAME, specified in the Access File, and establishes the starting position at the beginning (lowest value) of the key. Only one command is issued per request.

GETIT commands follow the GSETL command. They retrieve the element(s) for each root record indexed by the key. The data adapter will request only the elements necessary to satisfy the request. If there are no sort fields in the request, the answer set will be produced, in ascending order, by the key. GETIT commands will be issued repeatedly until DATACOM issues a return code of 19 (End of Table).

As long as the KEYNAME in the Access File is the Native Key, the data adapter will retrieve all records which correspond to a DATACOM table.

SELFR and SELNR

The data adapter uses two types of selection criteria to construct the SELFR call to DATACOM for a record:

1. All the values supplied in WHERE statements that mention a field in the segment and have any of these types of operators:

EQ	LT
IS	NE
GE	CONTAINS
GT	FROM . . . TO
LE	

Note: The SELFR request does not use selection on defined fields.

2. DBA value selection criteria on the segment.

The data adapter generates a SELFR command, using all available selection criteria on the segment. This builds a list of records which match the selection criteria and returns the first record. The list is built in the temporary index area of the DATACOM data source.

For a descendant record with SEGTYPE=U, the SELFR retrieves the unique descendant. No SELNR command is issued. Otherwise, the SELFR command is followed by SELNR command(s) which retrieve the records listed in the temporary index. A SELNR is issued repeatedly until DATACOM issues a return code of 14 (End/Beginning of Set).

To illustrate the sequence of calls the data adapter must make to DATACOM to service a multi-segment request, we will use the PERSPAY data source.

```
FILENAME=PERSPAY , SUFFIX=DATACOM
SEGNAME=PERSON , SEGTYPE=S , $
FIELD=EMP_NO , ALIAS=EMDTA.EN , USAGE=P6 , ACTUAL=Z5 , $
FIELD=NAME , ALIAS=EMDTA.NAME , USAGE=A24 , ACTUAL=A24 , $
FIELD=STREET , ALIAS=EMDTA.STR , USAGE=A24 , ACTUAL=A24 , $
FIELD=CITY , ALIAS=EMDTA.CITY , USAGE=A15 , ACTUAL=A15 , $
FIELD=STATE , ALIAS=EMDTA.STAT , USAGE=A2 , ACTUAL=A2 , $
FIELD=ZIP , ALIAS=EMDTA.ZIP , USAGE=A5 , ACTUAL=A5 , $
FIELD=SSNO , ALIAS=EMDTA.SSNO , USAGE=P9L , ACTUAL=P5 , $
SEGNAME=PAYROLL , SEGTYPE=S , PARENT=PERSON , $
FIELD=ENUM , ALIAS=PYIDT.EN , USAGE=P6 , ACTUAL=Z5 , $
FIELD=ACTCODE , ALIAS=PYIDT.ACODE , USAGE=A1 , ACTUAL=A1 , $
FIELD=ACTSTATUS , ALIAS=PYIDT.ASTATU , USAGE=A1 , ACTUAL=A1 , $
FIELD=CURRATE , ALIAS=FIGSS.CRATE , USAGE=P10.2 , ACTUAL=Z8 , $
FIELD=YTDWAGES , ALIAS=FIGSS.YWAGES , USAGE=P10.2 , ACTUAL=Z8 , $
FIELD=YTDCOMM , ALIAS=FIGSS.YCOMM , USAGE=P10.2 , ACTUAL=Z8 , $
FIELD=YTDTAXES , ALIAS=FIGSS.YTAXES , USAGE=P10.2 , ACTUAL=Z8 , $
```

In the request:

```
SELECT NAME EMP_NO SSNO YTDWAGES YTDCOMM YTDTAXES
FROM PERSPAY
WHERE ACTSTATUS = 'A'
```

PERSON is a referenced segment; it is the root of the accessed subtree. NAME, EMP_NO, and SSNO are the names of fields on this segment.

However, there are no available selection criteria for the PERSON segment (ACTSTATUS is a field on the PAY segment). Therefore, the data adapter will first issue a GSETL command for the PAYROLL MASTER File (PMF), using the EMP_NO Native Key. The GSETL command is followed by a GETIT command to retrieve the first root record, the EMDTA element.

Since server always processes from top-to-bottom; left-to-right, all the related descendants of this first root record must be retrieved before proceeding to the next root record.

Next, to generate the SELFR call to retrieve PAY data source records related to the PERSON parent, two pieces of selection criteria will be used: the value of EMP_NO (the KEYFLD) from the PMF record, and the selection criteria on ACTSTATUS.

The first SELFR call to DATACOM for the first root record will retrieve a PAY record with ENUM equal to EMP_NO, and ACTSTATUS equal to 'A'. Subsequent SELNR calls (SEGTYPE=S) may retrieve other records with those same values.

After DATACOM returns a 14 for the PAY data source, a second GETIT command will be generated for a PMF record. The value of this record's EMP_NO field is used, with ACTSTATUS EQ A, to generate a new set of SELFR/SELNR calls to retrieve related PAY records.

This process will be repeated until DATACOM returns a 19 for the GETIT command on the root, signifying all records have been retrieved and processed. If the request were:

```
SELECT NAME EMP_NO SSNO YTDWAGES YTDCOMM YDPTAXES  
FROM PERSPAY  
WHERE ACTSTATUS = 'A'  
WHERE STATE = 'CT' OR 'RI' OR 'MA' OR 'VT' OR 'NH' OR 'ME'
```

SELEFR/SELNR commands would be issued, instead of GSETL/GETIT commands with the five STATE values (STATE is a field on the PERSON segment). The process however, would be the same.

Finally, keep in mind:

- The data adapter will retrieve all descendant records of one parent occurrence before it will retrieve the next parent record.
- The higher you put your selection criteria in the hierarchical structure, the more efficient processing will be.

CHAPTER 7

Getting Started in DB2

Topics:

- Preparing the DB2 Environment
- Configuring the Data Adapter for DB2
- Managing DB2 Metadata
- Customizing the DB2 Environment
- Calling a DB2 Stored Procedure Using SQL Passthru

The Data Adapter for DB2 allows applications to access DB2 data sources. The adapter converts data or application requests into native DB2 statements and returns optimized answer sets to the requesting program.

Preparing the DB2 Environment

The DB2 Data Adapter minimally requires the installation of the DB2 Client. The DB2 Client allows you to connect to a local or remote DB2 data source server.

For the server running on the OS/390 or Z/OS environment utilizing the CLI interface FMID JDB7717 DB2 component must be installed on the MVS system.

Procedure How to Set Up the Environment on Windows NT/2000

On Windows NT/2000, the DB2 environment is set up during the installation of DB2.

Procedure How to Set Up the Environment on UNIX

1. Specify the DB2 data source instance to access using the UNIX environment variable \$DB2INSTANCE. For example:

```
DB2INSTANCE=db2
export DB2INSTANCE
```

2. Specify the location of the DB2 instance you wish to access using the UNIX environment variable \$INSTHOME. For example, to set the home directory for the DB2 software to /usr/db2710, specify:

```
INSTHOME=/usr/db2710
export INSTHOME
```

3. Specify the path to the DB2 shared library using the UNIX environment variable \$LD_LIBRARY_PATH. For example:

```
LD_LIBRARY_PATH=$INSTHOME/sql/lib/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH
```

Procedure How to Set Up the Environment on OS/390 and z/OS

Two types of Data Adapters for DB2 are available in this environment. Each one requires different pre-configuration steps prior to using the Web Console.

- **DB2/CAF:** Utilizes embedded SQL via the IBM/DB2 for OS/390 Call Attach Facility to access the DB2 RDBMS. The DB2 Bind job called GENIDB2 must be submitted. This member of the *qualif.release.servertype.DATA* PDS is generated by the ISETUP procedure. See the *Server Installation Guide for OS/390 and z/OS* for specific information regarding this job.

Note: The GENIDB2 JCL will automatically be submitted by ISETUP if the DB2/CAF option is selected from within that procedure.

- **DB2/CLI:** Utilizes IBM/DB2 for OS/390 Call Level Interface calls to access the DB2 RDBMS.

The DB2/CLI adapter makes use of DB2 Plan DSNACLI, ensure that the GRANT EXECUTE command is issued for all users who will be executing this plan.

Procedure How to Set Up the Environment on OS/400

To create tables with one-part names, a CURLIB must be set. Thereafter, the location can be anywhere in the library path. The first table found is the one used.

Accessing a Remote Database Server

Using the standard rules for deploying the DB2 Client, the server supports connections to:

- Local DB2 data source servers.
- Remote DB2 data source servers.

When using DB2/CLI to connect to a remote DB2 data source, the DB2 client catalog must contain an entry for the node where the remote data source resides and an entry for the remote database name.

When using DB2/CAF to connect to a remote DB2 data source, the DB2 Data Adapter supports IBM's Distributed Relational Database Architecture (DRDA):

- The Level 1 DRDA CONNECT command (OS/390 and z/OS).
- The Level 2 DRDA commands included in DB2 Version 3 and above.
- The SET CURRENT PACKAGESET command included in DB2 Version 2 Release 3 (OS/390 and z/OS) and above.

XA Support

Read/write applications accessing DB2 data sources are able to perform transactions managed in XA-compliant mode.

To activate the XA Transaction Management feature, the server has to be configured in Transaction Coordination Mode, using the Web console configuration functions. Using Transaction Coordination Mode guarantees the integrity of data modification on all of the involved DBMSs and protects part of the data modifications from being committed on one DBMS and terminated on another.

For complete documentation on XA compliance, see the XA Support appendix.

Configuring the Data Adapter for DB2

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to an DB2 database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Manually add the command in the global server profile (edasprof.prf) or in a user profile (user.prf).
- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the global server profile. For more information, see *Declaring DB2 Connection Attributes from the Web Console*.

You can declare connections to more than one DB2 database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to DB2 Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Syntax

How to Declare Connection Attributes Manually

Explicit authentication. The user ID and password are explicitly specified for each connection and passed to DB2, at connection time, for authentication. The syntax is

```
ENGINE [DB2] SET CONNECTION_ATTRIBUTES [data source]/user_ID,password
```

Password passthru authentication. The user ID and password are explicitly specified for each connection and passed to DB2, at connection time, for authentication. This option requires that the server be started with security off. The syntax is

```
ENGINE [DB2] SET CONNECTION_ATTRIBUTES [data source]/
```

Trusted authentication. The adapter connects to DB2 as a Windows login using the credentials of the Windows user impersonated by the server data access agent. The syntax is

```
ENGINE [DB2] SET CONNECTION_ATTRIBUTES [data source]/,
```

where:

DB2

Indicates the Data Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

data source

Is the name of the DB2 data source you wish to access.

user_ID

Is the primary authorization ID by which you are known to DB2.

password

Is the password associated with the primary authorization ID.

Example Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command allow the application to access the DB2 database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE DB2 SET CONNECTION_ATTRIBUTES SAMPLESERVER/MYUSER,PASS
```

The following SET CONNECTION_ATTRIBUTES command connects to the DB2 database server named SAMPLESERVER using Password Passthru authentication:

```
ENGINE DB2 SET CONNECTION_ATTRIBUTES SAMPLESERVER/
```

The following SET CONNECTION_ATTRIBUTES command connects to a local DB2 database server using operating system authentication:

```
ENGINE DB2 SET CONNECTION_ATTRIBUTES /,
```

Reference Declaring DB2 Connection Attributes From the Web Console

The DB2/CLI Adapter configuration screen displays the following fields:

Attribute	Description
Datasource	Is the user given name for the connection information. For OS/390 and z/OS, this is the DB2 location name as specified in the DB2 communications data source.

Attribute	Description
Security	<p>There are three methods by which a user can be authenticated when connecting to an DB2 database server:</p> <p>Explicit. The user ID and password are explicitly specified for each connection and passed to DB2, at connection time, for authentication.</p> <p>Password Passthru. The user ID and password received from the client application are passed to DB2, at connection time, for authentication. This option requires that the server be started with security off.</p> <p>Trusted. The adapter connects to DB2 as a Windows login using the credentials of the Windows user impersonated by the server data access agent.</p>
User	Is the authorization ID by which the user is known to DB2.
Password	Is the password associated with the user ID. The password is stored in encrypted form.

The DB2/CAF Adapter configuration screen displays the following fields:

Attribute	Description
SSID	DB2 SSID that is to be accessed, specified in the GENIDB2 job. This value must be upper case.
Plan	Plan that was bound to DB2 via the GENIDB2 job. This value must be upper case.
Execute DB2 BIND Command	Indicates whether to bind the program. Yes is the default value and adds the following fields to the configuration window: DSNCLST Library Name, Owner, Isolation Level.
DSNCLST Library Name	Is the name of the DSNCLST library. For example: DSN710.SDSNCLST This attribute is available only if you choose to bind your program.
Owner	Determines the authorization ID of the Owner of the Plan. This attribute is available only if you choose to bind your program
Isolation Level	Is the isolation level. The default value is CS. This attribute is available only if you choose to bind your program.

When you have completed the fields, click *Configure* to generate the adapter and build a DB2 plan.

Click *Grant* to grant plan execution to public.

DB2 CURRENT SQLID (OS/390 and z/OS)

DB2 accepts two types of IDs, the primary authorization ID and one or more optional secondary authorization IDs. It also recognizes the CURRENT SQLID setting.

Any interactive user or batch program that accesses a DB2 subsystem is identified by a primary authorization ID. A security system, such as RACF, normally manages the ID. During the process of connecting to DB2, the primary authorization ID may be associated with one or more secondary authorization IDs (usually RACF groups). Each site controls whether it uses secondary authorization IDs. For more information about using the data adapter in conjunction with external security packages, see the server manual for your platform.

The primary authorization ID is the same ID passed to the server at connect time. This user ID is then used to connect to the DB2 subsystem.

The DB2 Data Source Administrator may grant privileges to a secondary authorization ID that are not granted to the primary ID. Thus, secondary authorization IDs provide the means for granting the same privileges to a group of users. (The DBA associates individual primary IDs with a secondary ID and grants the privileges to the secondary ID.)

The DB2 CURRENT SQLID may be the primary authorization ID or any associated secondary authorization ID. At the beginning of the session, the CURRENT SQLID is the primary authorization ID.

Overriding the Default Connection

Once connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax How to Change the Default Connection

```
ENGINE [DB2] SET DEFAULT_CONNECTION [connection]
```

where:

DB2

Indicates the Data Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the service name defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following error message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following error message is issued: FOC1671, Command out of sequence.

Example Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the DB2 database server named SAMPLENAME as the default DB2 database server:

```
ENGINE DB2 SET DEFAULT_CONNECTION SAMPLENAME
```

Syntax How to Reset CURRENT SQLID

You can reset the CURRENT SQLID in a stored procedure or a profile using the following data adapter command

```
ENGINE [DB2] SET CURRENT SQLID = 'sqlid'
```

where:

DB2

Indicates the Data Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

sqlid

Is the primary or secondary authorization ID, which must be enclosed in single quotation marks as shown. All DB2 security rules are respected.

The CURRENT SQLID is the default owner ID for DB2 objects, such as tables or indexes, created with dynamic SQL commands. The CURRENT SQLID is also the sole authorization ID for GRANT and REVOKE commands. It must have all the privileges needed to create objects as well as GRANT and REVOKE privileges. In addition, the CURRENT SQLID is the implicit owner for unqualified table names.

Other types of requests, such as SQL SELECT, INSERT, UPDATE, or DELETE requests, automatically search for the necessary authorization using the combined privileges of the primary authorization ID and all of its associated secondary authorization IDs, regardless of the DB2 CURRENT SQLID setting.

The CURRENT SQLID setting remains in effect until the thread to DB2 is disconnected, when it reverts to the primary authorization ID.

Controlling Connection Scope (OS/390 and z/OS)

The SET AUTODISCONNECT and AUTOCLOSE commands control the persistence of connections when using the adapter.

Controlling Connection Scope with AUTOCLOSE

SET AUTOCLOSE initiates the DB2 Call Attachment Facility (CAF) CLOSE operation. It determines how long a thread (the connection between the application program in the user's address space and the DB2 application plan) is open. The thread is not the same as the address space connection to DB2; that connection is controlled by the AUTODISCONNECT setting.

In the server, an application program is one of the following:

- The dynamic DB2 Data Adapter.
- A CALLPGM subroutine using embedded SQL (non-CLI).

Generally speaking, each program has a corresponding plan or package.

A site that installs a DB2 subsystem determines the maximum number of concurrent users (threads) the subsystem will support. Since each user requires enough virtual storage for their application plan, this setting controls the amount of storage the site wants to allocate to active DB2 users at any one time.

The CAF CLOSE command de-allocates the DB2 thread, releasing the virtual storage for the application plan. DB2 requires that an existing thread to a plan be closed before a thread to another plan is opened. If a thread is closed without a subsequent OPEN operation, the closed thread becomes "inactive"; the user is still connected to DB2, but not to a particular application plan. The user (task) still owns the thread; it is not available to other users. To release the thread, the user must disconnect completely from DB2.

Note: The term pseudo-conversational describes the type of transaction processing provided when you use AUTOCLOSE ON COMMIT.

Controlling Connection Scope with AUTODISCONNECT

AUTODISCONNECT completely detaches the user's address space (or task) from DB2. This differs from CLOSE because after a CLOSE, the task is still connected to the DB2 subsystem and can open a thread to another plan. After a DISCONNECT, the task must reestablish its connection to DB2 before performing any data source work. The tasks that frequently issue the DISCONNECT command are connected to DB2 for shorter periods of time, allowing other tasks to connect and acquire threads as needed. However, there is significant system overhead associated with frequently connecting and disconnecting, and the possibility exists that no thread will be immediately available when the task attempts to reconnect.

Syntax

How to Control Connection Scope

```
ENGINE [DB2] SET {AUTOCLOSE|AUTODISCONNECT} ON {FIN|COMMIT}
```

where:

DB2

Indicates the Data Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

AUTOCLOSE

Issues the DB2 CAF CLOSE operation.

AUTODISCONNECT

Issues the DB2 Call Attach Facility (CAF) DISCONNECT operation.

FIN

Disconnects automatically only after the server session has been terminated. This value is the default.

COMMIT

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing DB2 Metadata

When the server accesses a data source, it needs to know how to interpret the data that it finds. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the DB2 data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each DB2 table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click Metadata. The Editing Metadata page opens.

To create a synonym, you must have configured the adapter. See *Configuring the Data Adapter for DB2* for more information.

2. Expand the *Add* folder, expand the adapter folder, and then click a connection. The right pane displays table and view selection options:

Select All Tables/Views. Select this radio button to create synonyms for all tables and views. This value is the default.

Filter by Name, Owner and Table Type. Select this radio button to filter the tables or views for which to create synonyms.

Selecting this option adds the following:

Select all Tables/Views

Filter by Name, Owner and Table Type

Owner - Sample: abc%

Table name - Sample: abc%

Table type TABLE VIEW TABLE/VIEW

Select Tables

Owner. Enter a string for filtering the owners' IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owners' IDs begin with the letters ABC; %ABC to select tables or views whose owners' IDs end with the letters ABC; %ABC% to select tables or views whose owners' IDs contain the letters ABC at the beginning, middle, or end.

Table Name. Enter a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

Table Type. Select one of the following options: Tables (this is the default), Views, Tables/Views.

3. Click *Select Tables*. All tables that meet the specified criteria are displayed:

Prefix: SET CONVERSION LONGCHAR:

Select Application Directory:

All Select

Check	Default Synonym Name	Owner/Schema	Table Name	Type
<input type="checkbox"/>	<input type="text" value="NF29001"/>	EDAQA	NF29001	TABLE
<input type="checkbox"/>	<input type="text" value="NF29002"/>	EDAQA	NF29002	TABLE
<input type="checkbox"/>	<input type="text" value="NF29003"/>	EDAQA	NF29003	TABLE
<input type="checkbox"/>	<input type="text" value="NF29004"/>	EDAQA	NF29004	TABLE
<input type="checkbox"/>	<input type="text" value="NF29005"/>	EDAQA	NF29005	TABLE
<input type="checkbox"/>	<input type="text" value="NF29006"/>	EDAQA	NF29006	TABLE
<input type="checkbox"/>	<input type="text" value="NF29007"/>	EDAQA	NF29007	TABLE
<input type="checkbox"/>	<input type="text" value="NF29008"/>	EDAQA	NF29008	TABLE
<input type="checkbox"/>	<input type="text" value="NF29009"/>	EDAQA	NF29009	TABLE
<input type="checkbox"/>	<input type="text" value="NF29012"/>	EDAQA	NF29012	TABLE
<input type="checkbox"/>	<input type="text" value="NF29013"/>	EDAQA	NF29013	TABLE

4. If you have tables with identical table names, assign a prefix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave field blank.

5. From the SET CONVERSION LONGCHAR drop-down list, select: Text, Alpha, or BLOB. The default value is Text. See *Controlling the Mapping of Large Character Data Types* for more information.
6. From the Select Application Directory drop-down list, select a directory. The default value is baseapp.
7. Complete your table or view selection:

To select all tables or views in the list, click **All**.

To select specific tables or views, click the corresponding check boxes.

8. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.
9. Click *Create Synonym*. Synonyms are created and added under the specified application directory.

Reference **Managing Synonyms**

In the navigation pane, click the name of the synonym to access the following options:

Sample Data	Retrieves up to 20 rows from the associated data source.
Properties	Displays a graphic representation of the synonym and enables you to edit its metadata.
Edit Master File	Enables you to manually edit the synonym's Master File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Master File, to update the synonym.
Edit Access File	Enables you to manually edit the synonym's Access File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Access File, to update the synonym.
Refresh	Regenerates the synonym. Use this option if structural changes were made to the data source.
Drop	Deletes the synonym.
Copy to	Copies the synonym to another application directory. Click the target directory from the resulting list.
Move to	Moves the synonym to another application directory. Click the target directory from the resulting list.

Syntax **How to Create a Synonym Manually**

```
CREATE SYNONYM app/synonym FOR table_view DBMS DB2 [NOCOLS]
END
```

where:

app

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, then this application name must not be used.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema) as follows:

```
[owner.] table
```

DB2

Indicates the Data Adapter for DB2.

NOCOLS

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.NF29004 DBMS DB2 AT DB1
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION,SUFFIX=DB2,$
SEGNAME=SEG1_4,SEGTYPE=S0,$
FIELD=DIVISION4,DIVISION4,I9,I4,MISSING=OFF,$
FIELD=DIVISION_NA4,DIVISION_NA4,A25,A25,MISSING=ON,$
FIELD=DIVISION_HE4,DIVISION_HE4,I9,I4,MISSING=ON,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,tablename=EDAQA.NF29004,
CONNECTION=DB1,KEYS=1,WRITE=YES,$
```

Reference Access File Keywords

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Is the name of the table or view. This value can include a location or owner name as follows: TABLENAME= [location.][owner.]tablename Note: Location is valid only with DB2/CAF and specifies the subsystem location name.
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=connection CONNECTION='' indicates access to the local DB2 database server. Absence of the CONNECTION attribute indicates access to the default database server.
DBSPACE	Optional keyword that indicates the storage area for the table. For example: datasource.tablespace DATABASE datasource
KEYS	Indicates how many columns constitute the primary key for the table. Range is 0 to 64. Corresponds to the first n fields in the Master File segment.
WRITE	Specifies whether write operations are allowed against the table.

Keyword	Description
KEYORDER	Optional keyword that specifies the logical sort sequence of data by the primary key; it does not affect the physical storage of data. Used by the adapter for FST. and LST. operations in report requests.

Data Type Support

The following table lists how the server maps DB2 data types. Note that you can:

- Control the mapping of large character data types.
- Control the mapping of variable-length data types.
- Change the precision and scale of numeric columns.

Be sure to review those options carefully as they affect the mapping of data types.

DB2 Data Type	Data Type		Remarks
	Usage	Actual	
CHAR (<i>n</i>)	<i>An</i>	<i>An</i>	<i>n</i> is an integer between 1 and 254
VARCHAR (<i>n</i>)	<i>An</i> TX50	<i>An</i> TX	<i>n</i> is an integer between 1 and 32672
BLOB	BLOB	BLOB	Up to 2 gigabytes
CLOB	TX50	TX	Up to 2 gigabytes
SMALLINT	I6	I4	
INTEGER	I11	I4	Maximum precision is 11
BIGINT	P20	P10	Available on UNIX & NT only
DECIMAL (<i>p,s</i>)	P6	P8	<i>p</i> is an integer between 1 and 31 <i>s</i> is an integer between 0 and <i>p</i>
REAL	F9.2	F4	Maximum precision is 9
FLOAT	D20.2	D8	Maximum precision is 20
DATE	YYMD	DATE	
TIME	HHIS	HHIS	
TIMESTAMP	HYYMDm	HYYMDm	Supported as Read-only

DB2 Data Type	Data Type		Remarks
	Usage	Actual	
LONG VARCHAR (<i>n</i>) in (1...32700)			Not supported
GRAPHIC			Not supported
VARGRAPHIC			Not supported
LONG VARGRAPHIC			Not supported
DATALINK			Not supported

Controlling the Mapping of Large Character Data Types

The SET parameter CONVERSION LONGCHAR controls the mapping of the DB2 data type VARCHAR. By default, the server maps this data type as alphanumeric (A). The server data type A supports a maximum of 4096 characters for TABLE/MODIFY and 32768 characters for API applications.

The following table lists data type mappings based on the value of LONGCHAR:

DB2 Data Type	Remarks	LONGCHAR ALPHA or BLOB		LONGCHAR TEXT	
		Usage	Actual	Usage	Actual
VARCHAR (<i>n</i>)	<i>n</i> is an integer between 1 and 32768	<i>An</i>	<i>An</i>	TX50	TX

Syntax **How to Control the Mapping of Large Character Data Types**

```
ENGINE [SQLDB2] SET CONVERSION LONGCHAR {ALPHA|TEXT|BLOB}
```

where:

SQLDB2

Indicates the Data Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

ALPHA

Maps the DB2 data type VARCHAR as alphanumeric (A). This value is the default.

TEXT

Maps the DB2 data type VARCHAR as text (TX). Use this value for WebFOCUS applications.

BLOB

Is equivalent to ALPHA. That is, it maps the DB2 data type VARCHAR as alphanumeric (A).

Controlling the Mapping of Variable-Length Data Types

The SET parameter VARCHAR controls the mapping of the DB2 data types VARCHAR. By default, the server maps this data type as alphanumeric (A).

The following table lists data type mappings based on the value of VARCHAR:

Microsoft SQL Server Data Type	Remarks	VARCHAR ON		VARCHAR OFF	
		Usage	Actual	Usage	Actual
VARCHAR (<i>n</i>)	<i>n</i> is an integer between 1 and 32768	AnV	AnV	An	An

Syntax **How to Control the Mapping of Variable-Length Data Types**

```
ENGINE [SQLDB2] SET VARCHAR {ON|OFF}
```

where:

SQLDB2

Indicates the Data Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Maps the DB2 data type VARCHAR as variable-length alphanumeric (AnV).

OFF

Maps the DB2 data type VARCHAR as alphanumeric (A). This value is the default.

BLOB Activation (OS/390 and z/OS)

DB2 data types that support the 'for bit data' attribute including VARCHAR(n) where n > 256 and LONG VARCHAR, can be supported in the server as BLOBs (Binary Large Objects). This support is for both read and write access.

Syntax **How to Activate BLOB**

To activate this support, you must issue the following command in the one of the supported server profiles

```
ENGINE [DB2] SET CONVERSION LONGCHAR BLOB
```

where:

DB2

Indicates the DB2 Data Adapter. You can omit this value if you previously issued the SET SQLENGINE command.

BLOB

Activates long binary support. The default is ALPHA.

BLOB Read/Write Support (OS/390 and z/OS)

For DB2 data types VARCHAR (>256) and LONG VARCHAR which have the *for bit data* attribute, the server provides read and write support using three server remote procedures routines. These routines are:

EDABS	Used to send binary image data to the server.
EDABE	Used to mark the end of the binary image.
EDABK	Used to purge the binary image from server storage.

1. The sequence of operations for the client application is: Converts every binary byte of the image into 2 bytes of hexadecimal data and stores the result in an internal buffer. If the image is large, this could be split into manageable pieces.
2. Sends the converted binary bytes to the server using remote procedure EDABS. The server converts the hex data back to binary and stores the image ready for INSERT/UPDATE into DB2.
3. Repeats steps 1 and 2 until the complete image is sent to the server in hex format. It then sends remote procedure EDABE to mark the end of the image.
4. The client application prepares an SQL INSERT/UPDATE using parameter markers for the columns of the row.
5. The client application issues a BIND for the column using CHAR(16) for the image column.
6. The client application issues an EXECUTE USING command giving the data values for the row columns but using 'BLOB ' for the image. The row will be INSERTed/UPDATED using the image buffer stored on the server.
7. Once the application has finished with the stored image on the server (and COMMITed the data), it should send the EDABK Remote Procedure to release server storage.

For full details and examples of how to maintain DB2 'for bit data' columns, see the *API Reference* and *Connector for ODBC* manuals.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to a server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Syntax

How to Change the Precision and Scale of Numeric Columns

```
ENGINE [DB2] SET CONVERSION RESET
ENGINE [DB2] SET CONVERSION format RESET
ENGINE [DB2] SET CONVERSION format [PRECISION pp [ss]]
ENGINE [DB2] SET CONVERSION format [PRECISION MAX]
```

where:

DB2

Indicates the Data Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

RESET

Returns any previously specified precision and scale values to the data adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

format

Is any valid format supported by the data source. Possible values are:

INTEGER

Indicates that the command applies only to INTEGER.

DECIMAL

Indicates that the command applies only to DECIMAL columns.

REAL

Indicates that the command applies only to single precision floating point columns.

FLOAT

Indicates that the command applies only to double precision floating point columns.

pp

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

SS

Is the scale. This is valid with DECIMAL, REAL, and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. If the scale is not required, you must set mm to 0 (zero).

MAX

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
REAL	9
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example **Setting the Precision and Scale Attributes**

The following example shows how to set the precision attribute for all INTEGER fields to 7:

```
ENGINE DB2 SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE DB2 SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER fields to the default:

```
ENGINE DB2 SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE DB2 SET CONVERSION RESET
```

Customizing the DB2 Environment

The Data Adapter for DB2 provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Improving Response Time

If you are using the Data Adapter for DB2 on OS/390 or z/OS, note that DB2 Version 3 supports parallel query I/O and Version 4 supports parallel query CPU to improve response. The data adapter supports parallel processing if you issue the SET CURRENT DEGREE command prior to the request.

Syntax **How to Improve Response Time**

```
ENGINE [DB2] SET CURRENT DEGREE { '1' | 'ANY' }
```

where:

DB2

Indicates the Data Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

1

Invokes serial processing. This value is the default.

ANY

Invokes parallel processing for dynamic requests. If the thread to DB2 is closed during the session, the value resets to 1.

Designating a Default Tablespace

You can use the SET DBSPACE command to designate a default tablespace for tables you create. For the duration of the session, the adapter places these tables in the DB2 tablespace that you identify with the SET DBSPACE command. If the SET DBSPACE command is not used, DB2 uses the default tablespace for the connected user.

Syntax How to Designate a Default Storage Space for Tables

```
ENGINE [DB2] SET DBSPACE {datasource.tablespace|DATABASE datasource}
```

where:

DB2

Indicates the Data Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

datasource

Is the data source name. The default value is DSNDDB04, a public data source.

tablespace

Is a valid table space name in the data source.

Note: This command will only affect CREATE FILE requests issued by Table Services. It does not affect Passthru CREATE TABLE commands.

Specifying the Block Size for Array Retrieval

The Data Adapter for DB2 supports array retrieval from result sets produced by executing SELECT queries or stored procedures. This technique substantially reduces network traffic and CPU utilization.

Using high values increases the efficiency of requests involving many rows, at the cost of higher virtual storage requirements. A value higher than 100 is not recommended because the increased efficiency it would provide is generally negligible.

Syntax **How to Specify the Block Size for Array Retrieval**

The block size for a SELECT request applies to TABLE FILE requests, MODIFY requests, MATCH requests, and DIRECT SQL SELECT statements.

```
ENGINE [DB2] SET FETCHSIZE n
```

where:

DB2

Indicates the Data Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is the number of rows to be retrieved at once using array retrieval techniques. Accepted values are 1 to 5000. The default is 20. If the result set contains a column that has to be processed as a CLOB or a BLOB, the fetchsize value used for that result set is 1.

Controlling the Types of Locks

You can use the SET ISOLATION command to specify the isolation level of transactions created by the adapter. The isolation level controls the types of locks for objects referenced in the requests executed within the transaction.

Syntax How to Control the Lock Type

```
ENGINE [DB2] SET ISOLATION level
```

where:

DB2

Indicates the DB2 Data Adapter. You can omit this value if you previously issued the SET SQLENGINE command.

level

Sets the DB2 isolation level which is mapped to the IBM RDBMS isolation level. If you do not specify an isolation level, the level is reset to the data adapter default.

DB2 Isolation Level (CLI)	IBM RDBMS Isolation Level
RC (SQL_TXN_READ_COMMITTED)	CS (Cursor Stability)
SE (SQL_TXN_SERIALIZABLE_READ)	RR (Repeatable Read)
RR (SQL_TXN_REPEATABLE_READ)	RS (Read Stability)
RU (SQL_TXN_READ_UNCOMMITTED)	UR (Uncommitted Read)

RC releases shared locks as the cursor moves on in the table. Use for read-only requests. This value is the default. Maps to **CS** (*Cursor Stability*).

SE locks the retrieved data until it is released by an SQL COMMIT WORK or SQL ROLLBACK WORK statement. Maps to **RR** (*Repeatable Read*).

RR maps to **RS** (*Read Stability*). For more information, see the *DB2 Command and Utility Reference*.

RU provides read-only access to records even if they are locked. However, these records may not yet be committed to the data source. Maps to **UR** (*Uncommitted Read*).

Note:

- The data adapter does not validate the isolation level values. If you issue the SET ISOLATION command with an invalid value, the adapter will not return a message.
- To display the isolation level setting, issue the ENGINE [DB2] ? CONNECTINFO query command.

Overriding Default Parameters for Index Space

You can use the SET IXSPACE command to override the default parameters for the DB2 index space implicitly created by the CREATE FILE and HOLD FORMAT SQLORA commands.

Syntax **How to Set IXSPACE**

```
ENGINE [DB2] SET IXSPACE [index-spec]
```

where:

DB2

Indicates the Data Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

index-spec

Is the portion of the DB2 CREATE INDEX statement that defines the parameters for the index. It can consist of up to 94 bytes of valid Oracle index space parameters. To reset the index space parameters to their default values, issue the SET IXSPACE command with no parameters.

Note: Refer to the DB2 documentation for more information on this command.

The long form of SQL Passthru syntax for commands exceeding one line is:

```
ENGINE DB2  
SET IXSPACE index-spec  
END
```

For example, to specify the NOSORT, NOLOGGING, and TABLESPACE portions of the DB2 CREATE INDEX statement, enter the following commands:

```
ENGINE DB2  
SET IXSPACE NOSORT NOLOGGING  
TABLESPACE TEMP  
END
```

Note: This command will only affect CREATE INDEX requests issued by CREATE FILE and HOLD FORMAT DB2 commands. It does not affect Passthru CREATE INDEX commands, for example:

```
ENGINE DB2 SET IXSPACE TABLESPACE tablespace_name  
TABLE FILE table_name  
PRINT *  
ON TABLE HOLD AS file_name FORMAT DB2  
END
```

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Syntax How to Obtain the Number of Rows Updated or Deleted

```
ENGINE [DB2] SET PASSRECS {ON|OFF}
```

where:

DB2

Indicates the Data Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. This value is the default.

OFF

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Using DB2 Aliases

You can use the SET REMOTECAT command to take advantage of DB2 aliases. DB2 allows a table alias in one DB2 subsystem to access a physical table in another DB2 subsystem. The physical table can be located in either a local or remote subsystem. You can include the command in any of the supported server profiles.

How to Use an Alias in One DB2 Subsystem to Access a Physical Table in Another

```
ENGINE DB2 SET REMOTECAT ON
```

Calling a DB2 Stored Procedure Using SQL Passthru

DB2 stored procedures are supported via SQL Passthru. These procedures need to be developed within DB2 using the CREATE PROCEDURE command.

Example Calling a Stored Procedure

The supported syntax to call a stored procedure is shown below. It is recommended that you use the syntax below instead of the previously supported CALLDB2 syntax.

```
ENGINE DB2  
EX SAMPLE PARM1 , PARM2 , PARM3 . . . ;  
TABLE ON TABLE PCHOLD  
  
END
```

The server supports invocation of stored procedures written according to the following rules:

- Only scalar input parameters are allowed, but not required.
- Output parameters are not allowed.
- The cursor must be opened in the procedure.
- No fetching is allowed in the stored procedure. The DB2 Data Adapter fetches the answer set.

Example Stored Procedure

```

CREATE PROCEDURE SAMPLE (IN l_name char(20))
  RESULT SETS 1
  LANGUAGE SQL
P1: BEGIN

  -- Declare cursor
  DECLARE cursor1 CURSOR WITH RETURN FOR
    SELECT
      EDAQA.NF29005.SSN5 AS SSN5,
      EDAQA.NF29005.LAST_NAME5 AS LAST_NAME5,
      EDAQA.NF29005.FIRST_NAME5 AS FIRST_NAME5,
      EDAQA.NF29005.BIRTHDATE5 AS BIRTHDATE5,
      EDAQA.NF29005.SEX5 AS SEX5
    FROM
      EDAQA.NF29005
    WHERE
      (
        ( EDAQA.NF29005.LAST_NAME5 = l_name )
      );

  -- Cursor left open for client application

  OPEN cursor1;
END P1

```

The following syntax is used to call the above stored procedure:

```

ENGINE DB2
EX SAMPLE ;
TABLE ON TABLE PCHOLD
END

```

Calling a DB2 Stored Procedure Using SQL Passthru

CHAPTER 8

Getting Started in Enterprise Java Beans

Topics:

- Preparing the Web Application Server Environment
- Configuring the Data Adapter for Enterprise Java Beans
- Managing Enterprise Java Beans Metadata

The Data Adapter for Enterprise Java Beans (EJB) allows applications to access Java Beans data sources. The adapter converts application requests into native Java Beans statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

The Data Adapter for Enterprise Java Beans currently supports access to the Entity Java Beans deployed under BEA WebLogic and IBM WebSphere Application Servers.

Preparing the Web Application Server Environment

The Web Application Server environment is set up during the installation of the corresponding Web Application Server or Client software.

Using the standard rules for deploying the Web Application Server or Client software, the server supports connections to:

- BEA WebLogic Application Server.
- IBM WebSphere Application Server.

Note: Enterprise Java Beans must be deployed on the Web Application Server.

Configuring the Data Adapter for Enterprise Java Beans

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

There are two methods by which a user can be authenticated when connecting to an Enterprise Java Beans database server:

- **Explicit.** User IDs and passwords are explicitly stated in SET CONNECTION_ATTRIBUTES commands. You can include these commands in the server global profile, edasprof.prf, for all users.
- **Password Passthru.** User IDs and passwords received from the client application are passed to the Web Application server for authentication.

When a client connects to the iWay Server, the user ID and password are passed to the Web Application Server for authentication, and are not authenticated by the iWay Server. To implement this type of authentication, start the iWay Server with security turned off. The server allows the client connection, then stores an encrypted form of the client's connection message. The encrypted message can be used for connection to a Web Application Server at any time during the duration of the server session.

Declaring Connection Attributes for the Web Application Server

The SET CONNECTION_ATTRIBUTES command allows you to declare a connection to one Web Application Server, and to supply the attributes necessary for connecting to the server.

You can connect to more than one Web Application Server by issuing multiple SET CONNECTION_ATTRIBUTES commands. The actual connection takes place when the first query referencing that connection is issued (see *Selecting a Web Application Server to Access* on page 8-5). You can include SET CONNECTION_ATTRIBUTES commands in an RPC or a server profile. The profile can be encrypted.

If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The first SET CONNECTION_ATTRIBUTES command sets the default Web Application Server database server to be used.
- If more than one SET CONNECTION_ATTRIBUTES command declares the same Web Application Server, the connection information is taken from the last SET CONNECTION_ATTRIBUTES command.

Syntax

How to Declare Connection Attributes for the Web Application Server

For BEA WebLogic Application Server with Explicit authentication:

```
ENGINE [EJB] SET CONNECTION_ATTRIBUTES connection_name
't3://hostname:port'/userid,password
```

For BEA WebLogic Application Server with Password Passthru authentication:

```
ENGINE [EJB] SET CONNECTION_ATTRIBUTES connection_name
't3://hostname:port'
```

For IBM WebSphere Application Server with Explicit authentication:

```
ENGINE [EJB] SET CONNECTION_ATTRIBUTES connection_name
'iiop://hostname:port'/userid,password
```

For IBM WebSphere Application Server with Password Passthru authentication:

```
ENGINE [EJB] SET CONNECTION_ATTRIBUTES connection_name
'iiop://hostname:port'
```

where:

EJB

Indicates the Data Adapter for Enterprise Java Beans.

connection_name

Is any name used as a connect descriptor to a Web Application Server across the network.

hostname

Is the host name of the computer where the Web Application Server is running.

port

Is the TCP/IP port number on which the Web Application Server is listening for incoming connections.

userid

Is the authorization ID known to the Web Application Server.

password

Is the password associated with the authorization ID.

Example **Declaring Connection Attributes for the BEA WebLogic Application Server**

The following SET CONNECTION_ATTRIBUTES command uses Password Passthru authentication to connect to BEA1, the BEA WebLogic Application Server running on the machine with hostname UNXSOL28:

```
ENGINE EJB SET CONNECTION_ATTRIBUTES BEA1 't3://UNXSOL28:7001'
```

The following SET CONNECTION_ATTRIBUTES command uses Explicit authentication to connect to MYBEA, a local BEA WebLogic Application Server:

```
ENGINE EJB SET CONNECTION_ATTRIBUTES MYBEA  
't3://localhost:7001'/USERA,PWDA
```

Example **Declaring Connection Attributes for the IBM WebSphere Application Server**

The following SET CONNECTION_ATTRIBUTES command uses Password Passthru authentication to connect to IBMCON1, the IBM WebSphere Application Server running on the machine with hostname AIX43:

```
ENGINE EJB SET CONNECTION_ATTRIBUTES IBMCON1 'iiop://AIX43:900'
```

The following SET CONNECTION_ATTRIBUTES command uses Explicit authentication to connect to MYWSP, a local IBM WebSphere Application Server:

```
ENGINE EJB SET CONNECTION_ATTRIBUTES MYWSP  
'iiop://localhost:900'/USERB,PWDB
```

Reference Declaring Connection Attributes from the Web Console

The Data Adapter for Enterprise Java Beans configuration screen displays the following fields:

Field	Description
Connection name	Enter an arbitrary connection name to be used to reference the connection.
URL	Enter the URL of the machine where the Web Application Server is running.
Security	There are two methods by which a user can be authenticated when connecting to an Java Beans database server: Explicit. The user ID and password are explicitly specified for each connection and passed to Java Beans, at connection time, for authentication. Password Passthru. The user ID and password received from the client application are passed to Java Beans, at connection time, for authentication. This option requires that the server be started with security off.
User	Enter the user name for the Web Application Server authenticated login.
Password	Enter the password that identifies the entered user name.

Selecting a Web Application Server to Access

After you have used the SET CONNECTION_ATTRIBUTES command to declare which Web Application servers will be accessed, select a specific Web Application Server from the list of declared servers. You can do this in one of two ways:

- Include the CONNECTION= attribute in the Access File of the table specified in the current EJB query. When you include a connection name in the CREATE SYNONYM command from the Web Console, the CONNECTION= attribute is automatically included in the Access File. This attribute supersedes the default connection.
- Do not include the CONNECTION= attribute in the Access File. The connection_name value specified in the *first* SET CONNECTION_ATTRIBUTES command is used.

Controlling Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections using the adapter for each of the connections you want to establish.

A connection occurs at the first interaction with the declared Web Application Server.

Syntax How to Control the Connection Scope

```
ENGINE EJB SET AUTODISCONNECT ON {FIN|COMMAND}
```

where:

FIN

Disconnects automatically only after the session has been terminated. This value is the default.

COMMAND

Disconnects automatically after each request. Depending on how often the event occurs, the AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server; it is related to the operating system and the Web Application Server.

Managing Enterprise Java Beans Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Java Beans data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Enterprise Java Bean (EJB) table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File, which represent the server's metadata.

Procedure How to Create a Synonym Using the Web Console

1. To create a synonym, click *Synonym, Create*. The Create Synonym window opens.

The Create Synonym window consists of two parts:

- A list of configured adapters and associated pre-configured connections.
- The selection criteria fields, which contain the EJB JNDI Name.

2. Select the connection for the Data Adapter for Enterprise Java Beans. Only one connection can be selected at any given time.
3. Enter the EJB JNDI Name. This must be the name of a node in the JNDI names tree.
4. After the selection has been made, click *Select EJB*. A window containing a list of EJB's that satisfy the selection criteria opens.

Note: Leaving selection fields blank on a production Web Application Server can produce a large list and take a long time to generate.

The Create Synonym window opens.

5. Select or enter the following when creating synonyms:

Parameter	Description
Select Application Directory	The application directory in which the synonyms will be created.

6. Select the EJBs for which you want to create synonyms; or select *All* and click *Create Synonym*. The Edit Synonyms window opens after the synonyms are created.

Procedure How to Modify a Synonym Using the Web Console

To modify a synonym file (including manual editing; testing and maintaining; copying and moving between application directories; and deleting):

1. Click *Synonym, Test*. The Edit Synonyms window opens.
2. Select the synonym and option.

From the Edit Synonyms window, you can:

Option	Description
Switch Current Application Directory	Changes the current application directory.
Edit Synonyms	Displays the file in editing mode when you click <i>name.mas</i> or <i>name.acx</i> .
Test Synonyms	Runs the selected request with a record limit of 50.
Drop/Drop All	Deletes selected or all metadata descriptions from the current application.
Copy/Copy All	Copies the selected synonyms or all synonyms from the current application directory to the selected application directory.
Move/Move All	Moves the selected synonyms or all synonyms from the current application directory to the selected application directory.

Syntax How to Create a Synonym Manually

```
CREATE SYNONYM appname/synonym FOR jndi_name DBMS EJB AT connection_name
END
```

where:

appname

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, this application name must not be used.

synonym

Is an alias for the EJB JNDI name (maximum 64 characters for Windows NT and UNIX Server platforms).

jndi_name

Is the JNDI name for the EJB.

EJB

Indicates the Data Adapter for Enterprise Java Beans.

AT *connection_name*

Is the connection name as previously specified in a SET CONNECTION_ATTRIBUTES command. When the server creates the synonym, this connection name becomes the value for CONNECTION= in the Access File.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note:

- CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.
- CREATE SYNONYM creates a Master File and Access File which represent the server metadata.

Example Using CREATE SYNONYM

Use the following syntax to create a synonym for the variable "Cities."

```
CREATE SYNONYM CITIES FOR CitiesEJB DBMS EJB AT BEACON1
END
```

CitiesEJB interfaces:

```
public interface CitiesEJBRemote extends javax.ejb.EJBObject {
    public String getData() throws RemoteException;
    public int getIndex() throws RemoteException;
    public void setData(int num) throws RemoteException;
    public void setIndex(int ind) throws RemoteException;
}

public interface CitiesEJBHomeRemote extends javax.ejb.EJBHome {
    public CitiesEJBRemote create()
        throws RemoteException, CreateException;
    public CitiesEJBRemote findByPrimaryKey(Integer primaryKey)
        throws FinderException, RemoteException;
}
```

Generated Master File cities.mas

```

FILENAME=CITIESG, SUFFIX=EJB      , $
$ =====
$ MFD generated for JNDI:CitiesEJB
$=====
    SEGMENT=PRIMARY_KEY, SEGTYPE=S0, $
$=====
$ constructors and parameters for PRIMARY_KEY:Integer
$=====
    GROUP=, ALIAS=K1, USAGE=A4, ACTUAL=A4, $
    FIELDNAME=K1F1, ALIAS=c0p0, USAGE=I11, ACTUAL=A4, $
    SEGMENT=OBJ01_CITIESEJBREMOTE, SEGTYPE=S0, PARENT=PRIMARY_KEY, $
$=====
$ OBJECT01, CitiesEJBRemote
$=====
    FIELDNAME=DATA, ALIAS=Data, USAGE=A256V, ACTUAL=A256V, MISSING=ON, $
    FIELDNAME=INDEX, ALIAS=Index, USAGE=I11, ACTUAL=A4, MISSING=ON, $

```

Generated Access File cities.acx

```

JNDINAME=CitiesEJB,
CONNECTION=MYCON,$
SEGNAME=PRIMARY_KEY,OBJECTNAME=Integer,$
SEGNAME=OBJ01_CitiesEJBRemote,OBJECTNAME=CitiesEJBRemote,$

```

The Master File root segment describes EJB Primary Keys. Each Primary Key is represented by a GROUP. Each KEY component is represented by an elementary FIELD in the GROUP. Only one active Primary Key is allowed in the request.

All other segments in the Master File describe the hierarchy of the OBJECTS that can be returned by EJB. Each GET Method of the OBJECT is represented by an elementary FIELD in the segment.

Reference Access File Keywords

Keyword	Description
JNDINAME	Identifies the EJB JNDI name.
CONNECTION= <i>connection_name</i>	Indicates access to the specified Web Application Server.
Absence of CONNECTION=	Indicates access to the default Web Application Server.

Data Type Support

The following chart provides information about the default mapping of Enterprise Java Beans data types to server data types:

EJB Data Type	Data Type		Remarks
	Actual	Usage	
VOID	A1	A1	
BYTE	A4	I4	
SHORT	A4	I6	
INT	A4	I11	
LONG	A20	P20	
FLOAT	A12	F12.2	
DOUBLE	A20	D20.2	
CHAR	A1	A1	
STRING	A256V	A256V	
BOOLEAN	A5	A5	

CHAPTER 9

Getting Started in Essbase

Topics:

- Preparing the Essbase Environment
- Configuring the Data Adapter for Essbase
- Managing Essbase Metadata
- Customizing the Essbase Environment

The Data Adapter for Essbase® allows applications to access Essbase data sources. The adapter converts application requests into native Essbase statements and returns optimized answer sets to the requesting application.

The Data Adapter for Essbase provides read-only access to Hyperion Cubes from a server running on Windows NT/2000. Essbase can only be accessed using an API supplied by the vendor. It is not possible to gain access using Direct SQL (Direct Passthru in server terminology) or the server's ODBC Data Adapter.

Preparing the Essbase Environment

The Data Adapter for Essbase minimally requires the installation of the Essbase Client. The Essbase Client allows you to connect to a local or remote Essbase Database Server.

Procedure How to Prepare the Environment on Windows NT/2000

On Windows NT/2000, the Essbase environment is set up during the installation of Essbase.

Procedure How to Prepare the Environment on UNIX

Using the UNIX environment variable \$ARBORPATH, specify the location of the Essbase Server you wish to access.

Example Specifying ARBORPATH on UNIX

To set the home directory for the Essbase Client to /usr/essbase, specify:

```
ARBORPATH=/usr/essbase  
export ARBORPATH
```

Configuring the Data Adapter for Essbase

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

When connecting to an Essbase Server, you must use Explicit authentication. This method requires that user IDs and passwords be explicitly stated in the SET CONNECTION_ATTRIBUTES commands. You can include these commands in the server global profile, edasprof.prf, for all users.

When you access an Essbase OLAP Server using the iWay Server, you are identified by a user ID and password. The Database Administrator can assign management privileges to a specific user, where privileges:

- Are constant for all applications and databases.
- Apply to an application.
- Apply to a database.

For more information on security, see the Hyperion Essbase documentation.

Syntax **How to Declare Connection Attributes Manually**

For Explicit authentication, use the syntax

```
ENGINE [ESSBASE] SET CONNECTION_ATTRIBUTES
[connection_name]/userid,password
```

where:

ESSBASE

Indicates the Data Adapter for Essbase. You can omit this value if you previously issued the SET SQLENGINE command.

connection_name

Is the server name used as a connect descriptor to the Essbase Server across the network.

userid

Is the primary authorization ID by which you are known to Essbase.

password

Is the password associated with the primary authorization ID.

Reference **Declaring Essbase Connection Attributes From the Web Console**

The Data Adapter for Essbase configuration screen displays the following fields:

Field	Description
Server	Enter the machine name where Essbase is running.
User	Enter the username by which you are known to Essbase.
Password	Enter the password associated with the username.

For details on how to declare connection attributes through the Web Console, see the *Server Configuration and Operation* manual for your platform.

Managing Essbase Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Essbase data type.

Creating Synonyms

Synonyms define unique names (or aliases) for each Essbase *application.database* combination that is accessible from a server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

For details on how to create a synonym through the Web Console, see the *Server Configuration and Operation* manual for your platform.

Syntax

How to Create a Synonym Manually

```
CREATE SYNONYM appname/synonym FOR application.database DBMS ESSBASE AT  
connection_name  
END
```

where:

appname

Is the 1 to 64 character application namespace where you want to create the synonym.
If your server is APP enabled, this application name must be used.

If your server is not APP enabled, then this application name must not be used.

synonym

Is an alias for one *application.database* combination.

application

Is the Essbase application that contains the database (Cube).

database

Is the database name (Cube) within the application.

ESSBASE

Indicates the Data Adapter for Essbase.

AT *connection_name*

Is the *connection_name* as previously specified in a SET CONNECTION_ATTRIBUTES command. When the server creates the synonym, this *connection_name* becomes the value for CONNECTION= in the Access File.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note:

- CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.
- The Master File that is produced contains a segment for each dimension in the Essbase outline. For each generation within a dimension, a field description is produced. The WITHIN keyword is used to describe the hierarchy inherent in each dimension structure.

Example Using CREATE SYNONYM (Master File)

Use the following syntax to create a synonym for the variable "sample".

```
CREATE SYNONYM sample FOR Sample.Basic DBMS ESSBASE AT UNXSOL26
END
```

Generated Master File sample.mas

```
FILENAME=SAMPLE, SUFFIX=ESSBASE, $
SEGMENT=BASIC, SEGTYPE=S0, $
```

```
$ DIMENSION: Scenario
```

```
FIELD=SCENARIO,
WITHIN=*Scenario,
ALIAS='Gen1,Scenario',
USAGE=A8,ACTUAL=A8, $
```

```
FIELD=GEN2_SCENARIO,
WITHIN=SCENARIO,
ALIAS='Gen2,Scenario',
USAGE=A10,ACTUAL=A10, $
```

Syntax **How to Use Default Field Names**

The default field names that are used are taken from the generation names, if available, in the outline; otherwise, they take the format

`FIELDNAME=generation_number_dimensionname`

where:

`generation_number`

Is the generation number within the dimension (for example, GEN2).

`dimensionname`

Is the name of the dimension.

For example, if the outline has a SCENARIO dimension, the first field name would be FIELDNAME=Scenario to represent the highest generation, the second field name would be FIELDNAME=GEN2_SCENARIO, and so on down the dimension hierarchy.

Example **Using CREATE SYNONYM (Access File)**

Generated Access File sample.acx

`SEGNAME=BASIC, SERVER=unxsol26, DBNAME=Basic, APPLNAME=Sample, $`

Reference **Access File Keywords**

Keyword	Description
<code>SERVER</code>	Identifies the Essbase Server name.
<code>DBNAME=database_name</code>	Indicates access to the specified database within an application.
<code>APPLNAME=application_name</code>	Indicates access to the specified application, which can contain one or more databases.

Parent/Child Support

The parent/child view in the Master File enables you to make requests using a member without having to know the generation to which the member belongs. The fields that provide this functionality are:

`DIMENSION_MEMBER`

Is the declaration for the field that contains the dimension members. It represents members of product at all levels of the dimension.

`DIMENSION_CAPTION`

Is the declaration for the field that contains the label displayed on reports for `DIMENSION_MEMBER` (this is the Essbase alias).

`DIMENSION_PARENT`

Is the declaration for the field that contains the parent of `DIMENSION_MEMBER`.

`DIMENSION_PARENTCAP`

Is the declaration for the field that contains the label displayed on reports for `DIMENSION_PARENT` (this is the Essbase alias).

The following is a sample of the parent/child view in the Master File:

```
FIELD=SCENARIO_MEMBER
WITHIN=*Scenario2
ALIAS=Scenario
USAGE=A10,ACTUAL=A10,$
FIELD=SCENARIO_CAPTION,
PROPERTY=CAPTION,REFERENCE=SCENARIO_MEMBER,
USAGE=A10,ACTUAL=A10,$
FIELD=SCENARIO_PARENT,
PROPERTY=PARENT_OF,REFERENCE=SCENARIO_MEMBER,
USAGE=A10,ACTUAL=A10,$
FIELD=SCENARIO_PARENTCAP,
PROPERTY=CAP_PARENT,REFERENCE=SCENARIO_MEMBER,
USAGE=A10,ACTUAL=A10,$
```

Example Using the Parent/Child View in the Master File

In the first request, ACTUAL is explicitly identified with the generation GEN2_SCENARIO:

```
TABLE FILE SAMPLE
PRINT PROFIT
WHERE GEN2_SCENARIO EQ Actual
END
```

In the second request, which takes advantage of parent/child view, ACTUAL is represented as a member of the Scenario dimension; it is not necessary to know which generation it falls under.

```
TABLE FILE SAMPLE
PRINT PROFIT
WHERE SCENARIO_MEMBER EQ Actual
END
```

Both requests yield the same output:

```
PROFIT = 105,522.00
```

Support for User-Defined Attributes (UDAs)

A User-Defined Attribute in Essbase enables you to select and report on data based on a common characteristic. You can include User-Defined Attributes in report requests. For each dimension with UDAs in an Essbase outline, CREATE SYNONYM generates a UDA field name under a segment called UDA in the Master File.

Example Reporting From a UDA Segment

The Sample Master File contains the following UDA segment:

```
$ DIMENSION: UDA
SEGMENT=UDA, SEGTYPE=U, PARENT=BASIC,$
FIELD=MARKET_UDA,
ALIAS=Market,
USAGE=A13,ACTUAL=A13, $
```

When referencing UDAs in a request, you must also reference a member of the dimension that contains the UDA. In this example, STATE is a member of the MARKET Dimension in the Master File.

```
TABLE FILE SAMPLE
PRINT SALES BY STATE
WHERE MARKET_UDA EQ 'New Market'
END
```

The output displays the values only for the states that have the UDA 'New Market' as a common characteristic:

STATE	SALES
-----	-----
Colorado	38,240.00
Louisiana	11,316.00
Nevada	29,156.00

Describing Scenario Dimensions in the Master File

Scenario dimensions in Essbase can be represented in two ways to the server: as *normal* dimensions or as *pseudo* account dimensions. If they are to be described as normal dimensions, the CREATE SYNONYM command generates field descriptions for the dimension. If you wish to describe them as pseudo in order to generate the Scenario dimension in place of the accounts dimension segment, use the SET SCENARIO command. The SET SCENARIO command must be executed before the synonym is created.

Syntax How to Generate a Pseudo Scenario Dimension

If you wish to generate the Scenario dimension to be used in place of the accounts dimension segment, use the syntax

```
SQL ESSBASE SET SCENARIO [DIM dimension_name] {member_name |ALL} FOR synonym
```

where:

dimension_name

Is the Scenario dimension name. The default name is Scenario. If the default name is used in the outline, DIM *dimension_name* can be omitted.

When used in the SET SCENARIO command, *dimension_name* is case-sensitive and must match the case in the Essbase outline.

member_name

Specifies a member name in the Scenario dimension to be used to generate a field for every account member intersection. When used in the SET SCENARIO command, *member_name* is case-sensitive and must match the case in the Essbase outline.

ALL

Indicates that all members of the Scenario dimension are used to generate the Master File. This is the default.

synonym

Is an alias for the data source (it can be a maximum of 64 characters).

Note: You can issue the SET SCENARIO command multiple times to specify that a number of members from the Scenario dimension be used in the generation of the Master File. Once this command has been issued, the CREATE SYNONYM command can be used to generate the Master File name.

For each Scenario/Accounts member intersection, a field is generated in the Master File. If this multiplicity effect causes the Master File that is generated to be invalid (due to the total length of the fields), the CREATE SYNONYM command fails and displays the following error message:

```
Total actual or usage exceeds 32768 To cut down, try SET SCENARIO
```

If you receive this message, you will need to issue fewer SET SCENARIO commands to limit the scope of the Master File with regards to the number of Scenario dimensions used.

Describing the Measures Dimension in the Master File

The SET MEASURE command enables you to set or change the Accounts tagged dimension without having to change the outline in the Essbase Database Server. With this setting, iWay produces an actual field name for every member of the named dimension in the Master File rather than producing the dimension in terms of generations. The SET MEASURE command must be executed before the synonym is created.

Syntax How to Set or Change the Measures Dimension in the Master File

To set the Accounts tag for a dimension, use the syntax

```
ENGINE ESSBASE SET MEASURE dimension_name FOR synonym
```

where:

dimension_name

Is the name of the dimension to be interpreted as an Accounts tagged dimension when generating a synonym.

synonym

Is an alias for one *application.database* combination.

Example Using SET MEASURE

The following command displays the actual member names of the Scenario dimension (for ACTUAL, BUDGET, or VARIANCE), rather than displaying fields like SCENARIO or GEN2_SCENARIO, in the Master File.

```
ENGINE ESSBASE SET MEASURE Scenario FOR SAMPLE
```


Reference Limiting the Number of Fields in a Master File

To address the 3,072 field limit in the Master File, SET MEASURE can be set to NONE. With this setting, all of the dimensions are interpreted as non-Accounts dimensions in the Master File and represented in terms of generations.

In addition, a segment called DATA is added to the Master File. This segment contains a field called DATA_VALUES, which enables you to display the values of the Measures dimension although the actual member names are not present in the Master File. This field should be used in conjunction with a GEN_MEASURES, MEASURE_MEMBER, or a generation field from the measures dimension.

Syntax How to Limit the Number of Fields in a Master File

```
ENGINE ESSBASE SET MEASURE NONE FOR SAMPLE
```

where:

NONE

Is the name of the dimension to be interpreted as an Accounts tagged dimension when generating a synonym.

Example Reporting Against a Master File in With a Limited Number of Fields

If SET MEASURE has been set to NONE, you can display the values of the Measures dimension although the actual member names are not present in the Master File. The following is a sample DATA segment, which includes the field DATA_VALUES, against which you can report.

```
$ DIMENSION: DATA
  SEGMENT=DATA ,SEGTYPE=U, PARENT=BASIC, $
  FIELD=DATA_VALUE, ALIAS=DATA_VALUE,
  USAGE=D20.2,ACTUAL=D8,MISSING=ON,
  TITLE=DATA_VALUE $
TABLE FILE SAMPLE
PRINT DATA_VALUE
BY PRODUCT
WHERE GEN2_MEASURE EQ 'Sales'
END
```

The output is:

```
PRODUCT          DATA_VALUE $
-----          -
```

PRODUCT	DATA_VALUE \$
Product	400,855.00

Changing the Default Usage Format of the Accounts Dimension

All dimension descriptions in the Master File, except for the accounts dimension, have the Usage format of alpha. The default format of the accounts dimension is D20.2. You can change this default using the SET CONVERSION command.

Syntax How to Change the Default Usage Format of the Accounts Dimension

```
ESSBASE SET CONVERSION format PRECISION n m
```

where:

format

Possible values are:

FLOAT which indicates that the command applies only to double precision floating point columns.

DECIMAL which indicates that the command applies only to decimal columns.

n

Is the precision. It must be a valid number representing the maximum value for precision for the data type.

m

Is a valid number representing the maximum value for scale for the data type.

If you do not specify a value for scale, the current scale setting remains in effect. If the scale is not required, you must set *m* to 0 (zero).

Customizing the Essbase Environment

The Data Adapter for Essbase provides several parameters for customizing the environment and optimizing performance. The following topics provide an overview of customization options.

Adapter Functionality

Data Adapter for Essbase functionality supports:

- JOINS.
- SUM and PRINT commands with Accounts/Measures dimensions.
- Reporting on parent/child views.

Reference JOIN Support

The Data Adapter for Essbase does not support direct JOINS from or to a cube. Therefore, SQL joins using a WHERE clause are not supported.

To achieve a joined request, use the FOCUS MATCH FILE command or create a HOLD file and use the hold file in the JOINed request.

Reference SUM and PRINT Support

The Essbase Server does not support the use of SUM or PRINT on an Accounts dimension member without reference to another non-Accounts dimension member in the request.

For example, since this request refers only to an Accounts dimension member,

```
TABLE FILE SAMPLE
PRINT SALES
END
```

the following message is displayed:

```
(FOC43244) No active dimension found in the request.
```

Reference Parent/Child View Support

- You cannot reference PARENT or PARENTCAP from the parent/child view without referencing MEMBER or CAPTION.

For example, the following request is incorrect since it does not reference MEMBER or CAPTION.

```
TABLE FILE SAMPLE
PRINT SALES
BY PRODUCT_PARENT
WHERE PRODUCT_PARENT EQ '100'
END
```

The following message displays:

```
(FOC43248) Can't reference PRODUCT's PARENT,PARENTCAP without MEMBER
or CAPTION.
```

The correct request is:

```
TABLE FILE SAMPLE
PRINT SALES
BY PRODUCT_PARENT
BY PRODUCT_MEMBER
WHERE PRODUCT_PARENT EQ '100'
END
```

The output is:

PRODUCT_PARENT	PRODUCT_MEMBER	Sales
100	100-10	62,824.00
	100-20	30,469.00
	100-30	12,841.00

- You cannot reference fields from the Generation view (GEN1_PRODUCT or GEN2_PRODUCT) and from the parent/child view (PRODUCT_MEMBER or PRODUCT_CAPTION) in one request if the fields from these different views are from the same dimension.

For example, in the following request

```
TABLE FILE SAMPLE
PRINT SALES
BY PRODUCT_MEMBER
BY FAMILY
WHERE PRODUCT_MEMBER EQ '100'
END
```

FAMILY is a member of the Product dimension and part of the Generation view so the following message is generated:

```
(FOC43247) Can't reference fields from Product2 and Product at the
same time.
```

The correct request is:

```
TABLE FILE SAMPLE
PRINT SALES
BY PRODUCT_MEMBER
BY SCENARIO
WHERE PRODUCT_MEMBER EQ '100'
END
```

The output is:

PRODUCT_MEMBER	SCENARIO	Sales
100	Scenario	106,134.00

Specifying ALIAS Names

In Essbase, you can assign more than one name to a member or a shared member. For example in the *Sample.Basic* database, the PRODUCT dimension contains members that can be identified by both product codes, such as 200, or by more descriptive names, such as COLA. By default, the output values for members specified in a request are the member values. The server allows you to take advantage of the more descriptive member names in the Essbase outline when formulating a request.

Syntax How to Specify ALIAS Names in a Request

```
ESSBASE SET USEALIASNAME {ON|OFF} [FOR synonym]
```

where:

ON

Allows the use of ALIAS names in a request.

OFF

Does not allow the use of ALIAS names in a request. This is the default setting.

synonym

Is an alias for the data source (it can be a maximum of 64 characters).

Note: If the SET USEALIASNAME command is used without FOR *synonym*, the setting applies to all Master Files. If the *synonym* option is used, the setting applies only to that Master File. You can issue multiple commands to control the use of the ALIAS name at the Master File level.

In an Essbase script, the reporting keyword OUTALTNAMES is equivalent to ALIAS.

Specifying ALIAS Tables

In Essbase, aliases are generally stored in one or more tables as part of the database outline. Once the SET command is active, screening conditions must use the ALIAS value not the member value.

If multiple ALIAS tables exist for an outline, you can use the SET ALIASTABLE command to specify which ALIAS table to use for a query.

Syntax How to Specify an ALIAS Table

To specify which ALIAS table to use for a query, use the syntax

```
ESSBASE SET ALIASTABLE {aliastablename|RESET} FOR synonym
```

where:

aliastablename

Is the ALIAS table name to be used.

RESET

Sets the alias table back to the current default for the outline in Essbase.

synonym

Is the Master File name.

Note: This SET USEALIASNAME command is used in conjunction with the SET USEALIASNAME command. If that command is not issued, the SET ALIASTABLE command is ignored. For more information, see *Specifying ALIAS Names* on page 9-15.

Setting the Maximum Number of Rows Returned

The default number of rows that can be returned from an Essbase cube using the server is 10,000. This can be controlled by the MAXROWS setting, which can be set in any supported server profile.

Syntax How to Set the Maximum Number of Rows Returned

To limit or extend the number of rows returned, use the syntax

```
SQL ESSBASE SET MAXROWS n [FOR synonym]
```

where:

n

Is a valid number that either limits or extends the number of rows to be returned. The default number of rows returned without this setting is 10,000. The maximum number for this setting is 999999999.

synonym

Is an alias for the data source (it can be a maximum of 64 characters).

Time Series Reporting

By using Time Series and Accounts tags within your Essbase outline, you can tell Essbase how to calculate your accounts data. When you tag a dimension as Time, Essbase knows that this is the dimension on which to base the time periods for the Accounts tags.

The server supports all 8 levels of period-to-date reporting within Essbase. See the *Hyperion Essbase Database Administrator's Guide* for further information.

In order to retrieve values from members in an Accounts dimension using time-series reporting, you must refer to at least one member of the Time dimension using the WHERE clause.

Example Using Time Series Reporting

```
SELECT PROFIT,PRODUCT,QTD FROM BASIC WHERE QTD = 'Mar' ORDER BY  
PRODUCT,QTD
```

Where PROFIT is a member of the Accounts dimension, PRODUCT is a member of a non-Accounts dimension and QTD is a member of the Time dimension. The above request returns the following row:

```
Product Q-T-D(Mar)                24703.00
```

Note: All results reference the Dynamic Time series member with dashes between letters.

Summing on Non-Aggregated Fields

Any two-pass calculated members that are found in the accounts (or Scenario) dimension and members with the (-), (\), (*) and (%) consolidation properties in the Essbase outline, have the following attributes in the associated Access File:

```
MEMBER=membername, AGGREGATE=NO, $
```

Due to the nature of two-pass calculation members, a summation request (SQL SUM... or SUM in TABLE) may produce incorrect values. Therefore, if the member has the AGGREGATE=NO attribute in the Access File, a SUM action in the request against the member results in the following error:

```
(FOC43241) Aggregation is requested for non aggregatable member(s)
```

The server allows SUM to be used on non-aggregated fields in the Access File.

Syntax

How to Turn Aggregation ON/OFF For Non-Aggregated Fields

```
SQL ESSBASE SET RESTRICTSUM [ON|OFF] FOR synonym
```

where:

ON

Does not allow the use of SUM on non-aggregated fields. This is the default setting.

OFF

Allows the use of SUM on non-aggregated fields.

synonym

Is an alias for the data source (it can be a maximum of 64 characters).

Example Using RESTRICTSUM on Non-Aggregated Fields

Using this Access File

```
SEGNAME=BASIC, SERVER=unxsol26, DBNAME=Basic, APPLNAME=Sample, $
TIMEDIM=Year, $
MEASURE=Measures, $
MEMBER=COGS, AGGREGATE=NO, $
MEMBER=TOTAL_EXPENSES, AGGREGATE=NO, $
MEMBER=PROFIT_%, AGGREGATE=NO, $
MEMBER=PROFIT_PER_OUNCE, AGGREGATE=NO, $
```

issue the following request:

```
TABLE FILE SAMPLE
SUM COGS
BY QUARTER
END
```

If RESTRICTSUM ON (default) is set in the profile, the following message is displayed and no output is generated:

```
(FOC43241) Aggregation is requested for non aggregatable member(s)
```

If you set RESTRICTSUM OFF in the profile, SUM is permitted on non-aggregated fields, producing this output:

QUARTER	COGS
Qtr1	42,877.00
Qtr2	45,362.00
Qtr3	47,343.00
Qtr4	43,754.00

Preventing Aggregation of Non-Consolidating Members

Member consolidation properties determine how children roll up into their parents in the Essbase outline. The members with the (~) as a consolidation property in the Essbase outline, are *not* rolled up in the database. The SET AGGREGATE NOOP (No Operation) setting only affects those Essbase members in the Accounts dimension with the (~) consolidation property. If AGGREGATE NOOP is set to OFF, SUM is not permitted on Essbase members with the (~) consolidation property and those members appear in the Access File as:

```
MEMBER=ADDITIONS, AGGREGATE=NO, $
```

Syntax **How to Prevent Aggregation of Non-Consolidating Members**

You must turn set AGGREGATE NOOP to OFF before you create the synonym.

The syntax is:

```
SQL ESSBASE SET AGGREGATE NOOP {ON|OFF}
```

where:

ON

Allows the use of SUM on NOOP Essbase members. With this setting, Essbase members are not be added to the Access File. This is the default setting.

OFF

Does not allow the use of SUM on NOOP Essbase members. Members with the (~) consolidation property are added to the Access File.

Note: Any member of the Accounts dimension in the Essbase outline tagged with (-), (/), (*) or (%) as a consolidation property automatically appears in the Access File with the AGGREGATE=NO setting.

Suppressing Shared Members

Shared Members in Essbase store pointers to data that is stored in the real member. Therefore, although the data is shared between the two members, it is only stored one time. You can exclude shared members from reports using the SUPSHARE setting.

Syntax **How to Suppress Shared Members in Essbase**

```
SQL ESSBASE SET SUPSHARE [ON|OFF]
```

where:

ON

Excludes shared members in a report.

OFF

Includes shared members in a report.

Example Suppressing Shared Members

The following request and output illustrates the affect of turning SUPSHARE OFF and ON.

```
TABLE FILE SAMPLE
PRINT SALES BY FAMILY BY SKU
AND COLUMN-TOTAL
WHERE FAMILY EQ 'Diet' OR '200'
END
```

With SUPSHARE OFF, the output is:

FAMILY	SKU	SALES
200	200-10	41,537.00
	200-20	38,240.00
	200-30	17,559.00
	200-40	11,750.00
Diet		
	100-20	30,469.00
	200-20	38,240.00
	300-30	36,969.00
=====		
TOTAL		214,764.00

With SUPSHARE ON, the output is:

FAMILY	SKU	SALES
200	200-10	41,537.00
	200-20	38,240.00
	200-30	17,559.00
	200-40	11,750.00
Diet		
	100-20	30,469.00
	300-30	36,969.00
=====		
TOTAL		176,524.00

Notice that member 200-20 is displayed twice in the first report, but only once in the second report. Turning SUPSHARE ON prevents a shared member from being duplicated in a report request.

Suppressing Zero Values

Essbase stores zero values that you can exclude from reports using the SUPERZEROS setting. This setting applies to an entire row. If one member in a row has a value of zero, and the other values in the same row have values other than zero, the zero value will not be suppressed. .

Syntax How to Suppress Rows With Zero Values

```
SQL ESSBASE SET SUPZEROS [ON|OFF]
```

where:

ON

Suppresses an entire row that contains 0 values.

OFF

Does not suppress zero values. This is the default setting.

Example Suppressing Zero Values

The following output illustrates the affect of turning SUPSHARE OFF and ON.

With SUPZEROS OFF, the output is:

STATE	SALES	PROFIT
Colorado	38,240.00	42,877.00
Louisiana	0.00	0.00
Nevada	29,156.00	47,343.00

With SUPZEROS ON, the output is:

STATE	SALES	PROFIT
Colorado	38,240.00	42,877.00
Nevada	29,156.00	47,343.00

Suppressing Missing Data

In Essbase, empty cells are known as missing or #MISSING data. You can suppress missing data during reporting using the SUPMISSING setting.

Syntax How to Suppress Rows With Missing Data

```
SQL ESSBASE SET SUPMISSING [ON|OFF]
```

where:

ON

Suppresses an entire row that contains #MISSING data.

OFF

Does not suppress rows with #MISSING data. This is the default setting.

Example Suppressing Missing Data

The following output illustrates the affect of turning SUPMISSING OFF and ON.

With SUPMISSING OFF, the output is:

STATE	SALES	PROFIT
Colorado	38,240.00	42,877.00
Louisiana	#MISSING	#MISSING
Nevada	29,156.00	47,343.00

With SUPMISSING ON, the output is:

STATE	SALES	PROFIT
Colorado	38,240.00	42,877.00
Nevada	29,156.00	47,343.00

Suppressing Zero Values and Missing Data

You can use the SUPEMPTY setting to suppress an entire row that contains either zero values, missing (#MISSING) data, or a combination of the two.

Syntax How to Suppress Rows With Zero Values or Missing Data

```
SQL ESSBASE SET SUPEMPTY [ON|OFF]
```

where:

ON

Suppresses an entire row that contains either zero values, #MISSING data, or a combination of the two.

OFF

Does not suppress rows that contain either zero values or #MISSING data. This is the default setting.

Example Suppressing Rows With Zero Values or Missing Data

The following output illustrates the affect of turning SUPEMPTY OFF and ON.

With SUPEMPTY OFF, the output is:

STATE	SALES	PROFIT
Colorado	38,240.00	42,877.00
Louisiana	#MISSING	#MISSING
Nevada	0.00	0.00

With SUPEMPTY ON, the output is:

STATE	SALES	PROFIT
Colorado	38,240.00	42,877.00

Substitution Variables

Substitution variables act as global placeholders for information that changes regularly. Each substitution variable configured in Essbase is assigned a value. You can change the value at any time, thus limiting the number of manual changes required in your scripts when you are running reports.

When you reference an Essbase substitution variable in a TABLE command, the substitution variable must be preceded with an up arrow (^) and enclosed in single quotation marks.

Example Using a Substitution Variable in a Request

In this example, CurrMonth is the name of the substitution variable that was set in Essbase. The ^ and quotation marks are required.

```
TABLE FILE SAMPLE
PRINT Product
BY
PRODUCT
WHERE
MONTH EQ '^CurrMonth'
END
```

CHAPTER 10

Getting Started in CA-IDMS/DB

Topics:

- Preparing the IDMS/DB Environment
- Configuring the Data Adapter for IDMS/DB
- IDMS/DB Overview and Mapping Considerations
- Managing IDMS/DB Metadata
- Master Files for IDMS/DB
- Access Files for IDMS/DB
- IDMS/DB Sample File Descriptions
- Customizing the IDMS/DB Environment
- Data Adapter for IDMS/DB Navigation
- IDMS/DB File Retrieval
- CA-IDMS/DB Record Retrieval Internals
- Tracing the Data Adapter for IDMS/DB

The Data Adapter for CA-IDMS/DB for OS/390 and z/OS allows applications to access IDMS/DB data sources. The adapter converts application requests into native IDMS/DB statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

Note: For the remainder of this manual, the terms

- CA-IDMS/DB
- IDMS/DB
- and
- Data Adapter for IDMS/DB

refer to the Data Adapter for CA-IDMS/DB.

Preparing the IDMS/DB Environment

Prior to configuring the Data Adapter for IDMS/DB using the Web Console, it is necessary to edit the ISTART JCL that is used to initiate the server.

Procedure How to Edit the ISTART JCL

1. Allocate the IDMS.LOADLIB and IDMS.DBA.LOADLIB libraries to the server's STEPLIB DDNAME allocation.
2. Allocate DDNAME SYSCTL to the IDMS.SYSCTL dataset.
3. Allocate DDNAME SYSIDMS to the IDMS.SYSIDMS dataset.

Configuring the Data Adapter for IDMS/DB

Configure the adapter using the Web Console Adapter Add screen and click *Configure*.

Declaring Connection Attributes

The integrity of the IDMS/DB data source is not jeopardized because the adapter provides read-only access. Access to the data source is restricted by the security package in use on the OS/390 system (such as RACF) and by the CA-IDMS/DB security features.

Both CA-IDMS/DB and the server provide security mechanisms to ensure that users have access to only those objects for which they have authorization.

IDMS/DB Overview and Mapping Considerations

CA-IDMS/DB databases, both network and LRF-based, correspond to hierarchical and relational data sources. The concepts discussed in this section affect the way IDMS/DB files are described to the server.

Mapping Concepts

IDMS/DB is a network database management system that is accessed by a subschema (the equivalent in the server is a Master File). IDMS/DB provides two methods of retrieving records within a subschema from an application program:

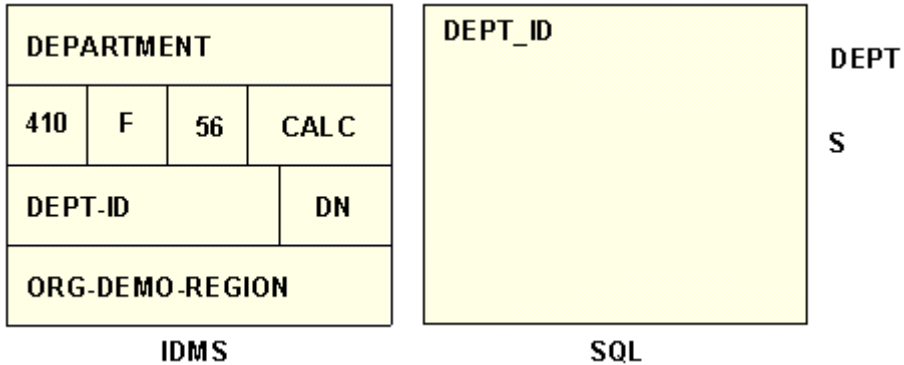
- DML (network access)
- LRF (LRF-based access)

DML (Data Management Language) access is the traditional method of IDMS/DB database navigation. It is the network navigation facility. Each physical record is retrieved separately. An application program enters the IDMS/DB database at a particular IDMS/DB record type (the equivalent in the server is a segment) and searches the set connections to retrieve the required data. For mapping concepts that apply to network record types, see *Summary of Network Relationships* on page 10-16.

LRF (Logical Record Facility) access, available as of IDMS/DB Release 5.7, is the relational-like method of access. LRF provides software to dynamically create flat records from one or more network record types at run time. For mapping concepts that apply to LRF-based records, see *Logical Record Facility Concepts* on page 10-14.

An IDMS/DB record type is described to the server as a segment. Since DML retrieval is record-oriented rather than field-oriented, the Master File must list the fields in the same order as they appear on the IDMS/DB record type. However, a Master File does not have to list every field of a particular record type; you may omit fields after a given field. For example, your description may list the first four fields of a 10-field IDMS/DB record type. You can use IDMS/DB field names in your Master File up to a maximum of 48 characters.

The following graphic illustrates how a record type is described as a segment. The record type DEPARTMENT contains the field-type DEPT-ID. Its corresponding server segment DEPT contains a field named DEPT_ID.



Repeating fields on an IDMS/DB record type are defined as OCCURS segments.

Network record types may be related to each other. These relationships may be physical (using set connections) or logical (achieved through an index or CALC field). The Data Adapter for IDMS/DB supports both physical and logical relationships.

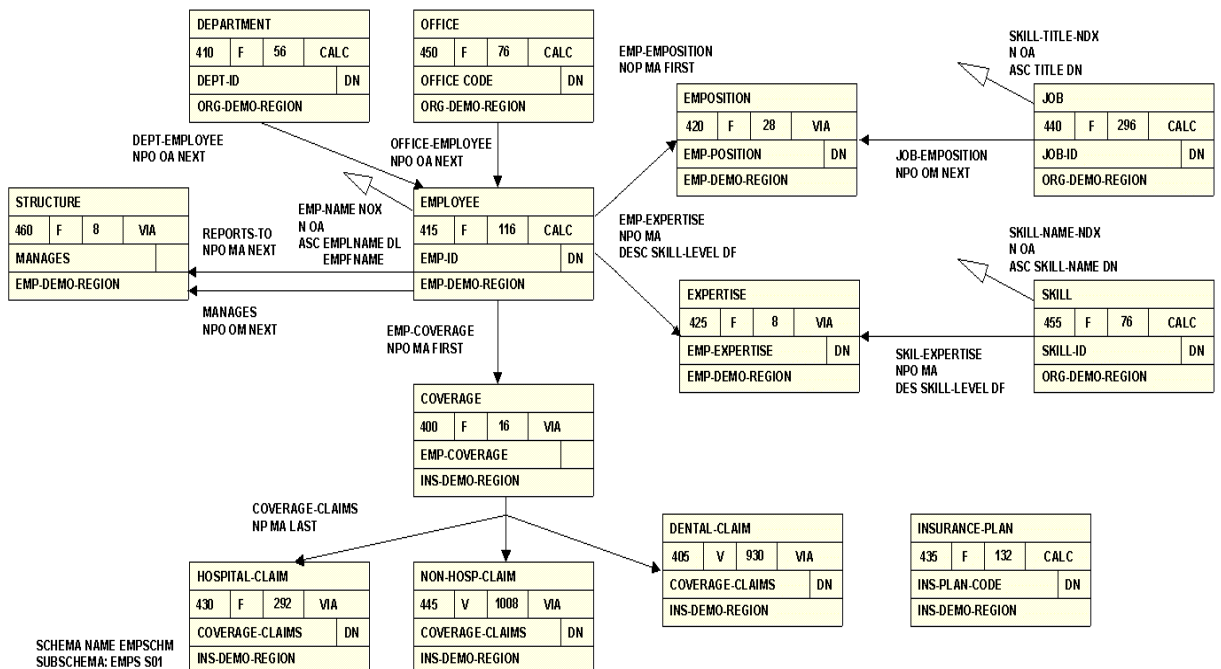
Set-Based Relationships

In an IDMS/DB database, physical relationships between record types are achieved with pointers that correspond to IDMS/DB sets. A set implements a one-to-many relationship between record types. The server equivalent of a set is the parent/descendant relationship between segments. In an IDMS/DB set, one record type acts as the owner (the one side of the relationship) and one or more record types act as the members (the many side of the relationship). A single IDMS/DB record type can participate in several set relationships as either the owner or the member.

The IDMS/DB representation of record types and set relationships within a database is called a Bachman diagram, also known as a data structure diagram. In a Bachman diagram, a record type is depicted as a box; a set, as a line with an arrow. Set names appear as labels beside the arrows. The box that the arrow points to is the member record type. Triangles indicate indexes. The next graphic is the Bachman diagram for the IDMS/DB network subschema EMPSS01. Sections of this diagram are referenced throughout these sections.

The following kinds of set-based relationships are depicted: simple set, common owner, common member, multi-member, bill-of-materials (simple and multi-tiered), and loop structures.

All relationships correspond to server structures and are explained following the diagram:



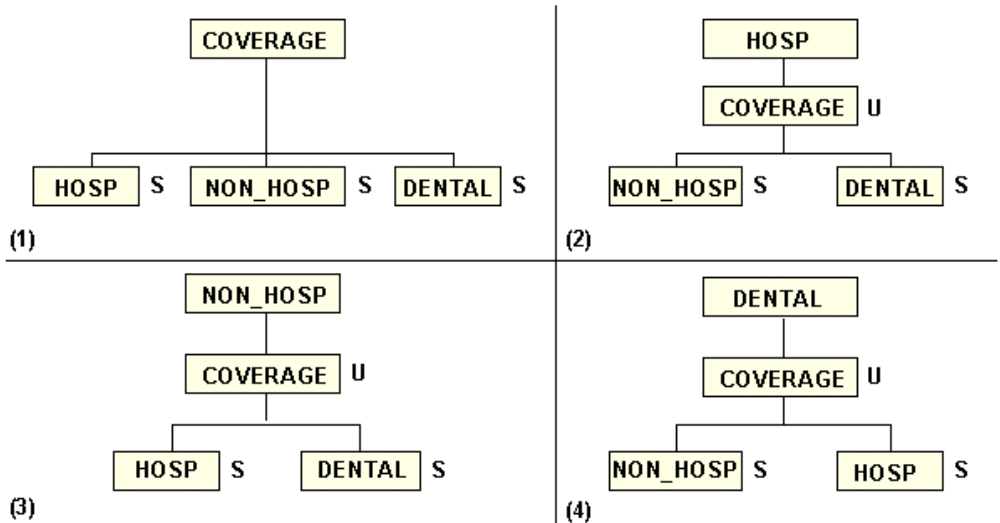
Simple Set

The graphic below illustrates the basic mapping principle of a simple set: an IDMS/DB record type corresponds to a segment; a set relationship corresponds to a parent/descendant relationship.

This figure also illustrates a second principle: an IDMS/DB structure can have more than one representation as a hierarchy. The type of parent/descendant relationship required in the Master File depends on whether the owner or the member record type is designated as the parent segment.

The JOB-EMPOSITION set has two possible representations:

1. It shows the JOB record type, the owner in the set, mapped to the server as the root segment. Since a member record type is multiply-occurring (for example, several instances of EMPOSITION records per JOB instance, indicating that many positions share one job title and description), the EMPOSITION record type is displayed as a non-unique descendant.
2. It depicts the reverse. The EMPOSITION record type is the root segment and JOB is the unique descendant, since an EMPOSITION instance can have only one owner (only one job title and description per position).

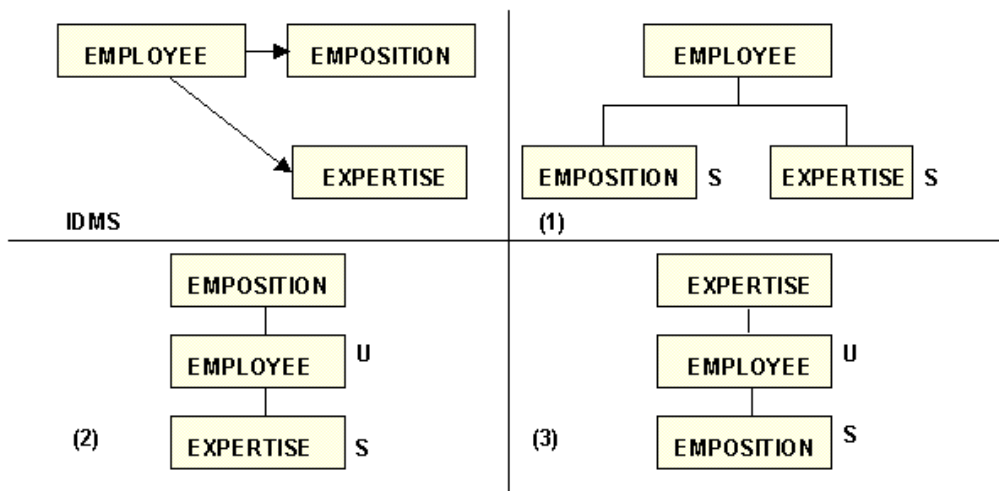


Common Owner

Given the rules in the previous example, consider a more complex scenario called a common owner. A common owner structure contains a record type that is the owner of two or more record types. Several representations are possible.

For example, the graphic below depicts three ways to describe the EMPLOYEE, EXPERTISE, and EMPOSITION structure in one Master File:

1. It shows the EMPLOYEE record type, the owner in both sets, mapped to the server as the root segment of EMPOSITION and EXPERTISE. Since an EMPLOYEE record can have many EMPOSITION and EXPERTISE records, both descendant records are non-unique.
2. It depicts the EMPOSITION record type as the root segment of EMPLOYEE, which acts as the parent of EXPERTISE. Since an EMPOSITION record can have only one owner, EMPLOYEE is a unique descendant; EXPERTISE is a non-unique descendant of EMPLOYEE.
3. It depicts the EXPERTISE record type as the parent of EMPLOYEE, which acts as the parent of EMPOSITION. EMPLOYEE is a unique descendant and EMPOSITION is a non-unique descendant of EMPLOYEE.

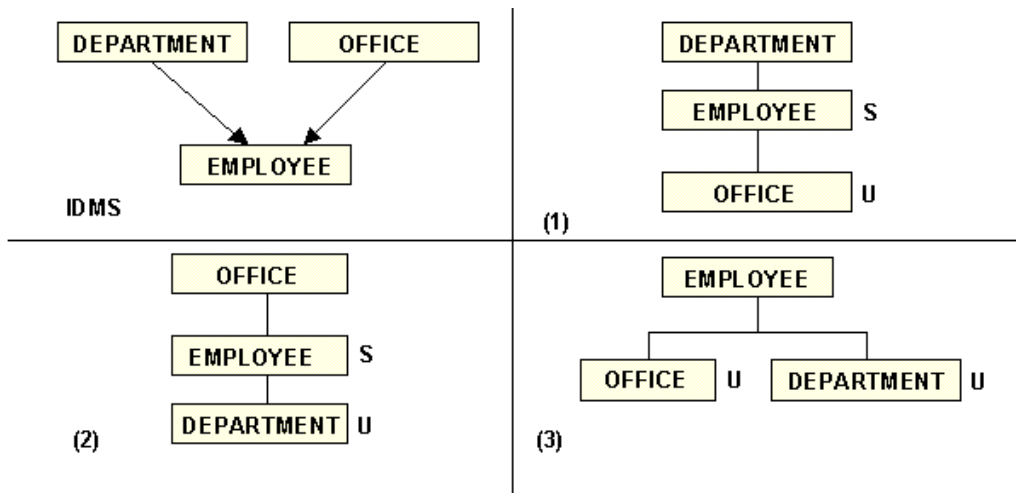


Common Member

When an IDMS/DB record type is a member of two or more sets, the association of the owner record type as the parent segment must be abandoned for one or more sets, because a segment can have only one parent. The graphic below displays the possible interpretations for this IDMS/DB configuration.

In the graphic, the EMPLOYEE record type is a common member in the DEPT-EMPLOYEE and the OFFICE-EMPLOYEE sets. This structure can be described to the server in three ways:

1. It shows DEPARTMENT as the root segment with EMPLOYEE as its non-unique descendant; and OFFICE is the unique descendant of EMPLOYEE.
2. It depicts the reverse: OFFICE is the root segment; EMPLOYEE is its non-unique descendant; and DEPARTMENT is the unique descendant of EMPLOYEE.
3. It shows the only other alternative: EMPLOYEE is the parent of OFFICE and DEPARTMENT. Both are unique, since an EMPLOYEE can belong to only one OFFICE and DEPARTMENT.



Notice that the rules for simple sets are still valid:

- If the owner record type is the parent segment, the member record type as a descendant segment is non-unique.
- If the member record type is the parent segment, the owner record type as a descendant segment is unique.
- A member or an owner record type may act as a root segment (the server ignores whether the root segment is unique or non-unique).

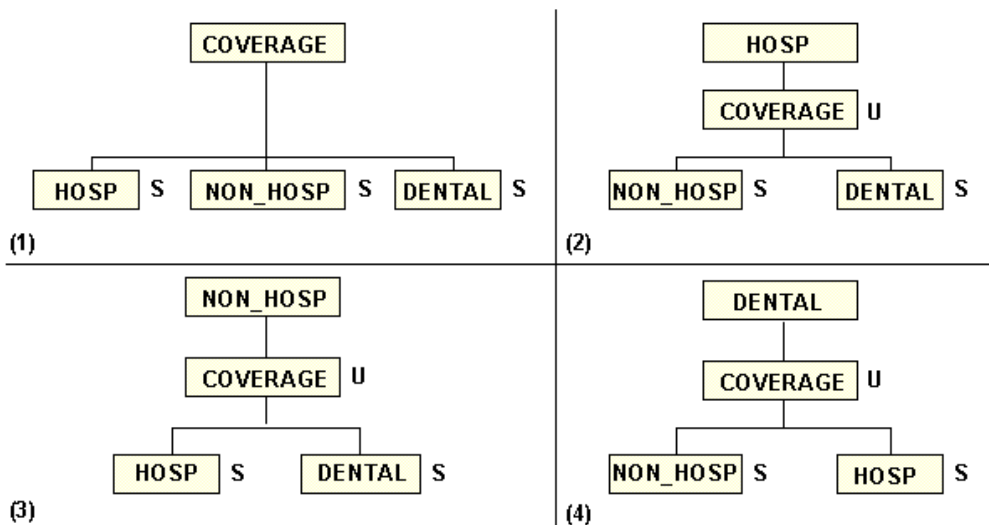
It may be helpful to think of the hierarchical depiction of a network structure as its navigational path. From an IDMS/DB standpoint, panel 1 of the above graphic shows that the IDMS/DB database can be entered at the DEPARTMENT record type. The corresponding EMPLOYEE record occurrences for a DEPARTMENT record occurrence can be obtained by searching the DEPT-EMPLOYEE set. For each EMPLOYEE record occurrence, obtaining the owner in the OFFICE-EMPLOYEE set retrieves the corresponding OFFICE record occurrence. This is a three-segment retrieval hierarchy that maps to server descriptions.

Multi-Member

When there is more than one member record type, the set is called a multi-member set. A multi-member set is represented in the Master File exactly like a common owner set. The fact that the two relationships are based on the same set is stated in the Access File.

For example, to describe the COVERAGE record type and its three members of the COVERAGE-CLAIMS set, you may choose one of four ways as depicted in the following graphic:

1. The first panel shows the COVERAGE record type, owner of the multi-member set, as the root segment. Since several instances of CLAIMS can be reported against one insurance policy (COVERAGE), each member is a non-unique descendant.
2. The second panel depicts HOSPITAL-CLAIM as the parent of COVERAGE, and the other two member record types as descendants of COVERAGE. In each case, a claim can be reported against only one insurance policy. This explanation applies to panels 3 and 4 as well.



Bill-of-Materials

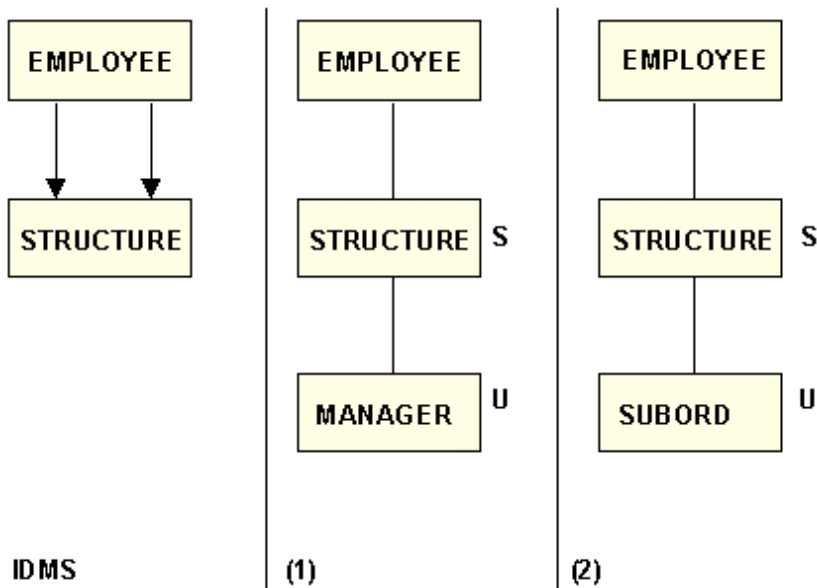
Bill-of-materials structures are classified as simple or multi-tiered. In this section, the simple version is discussed first.

An instance of two record types being linked by more than one set is called a bill-of-materials structure. This structure describes a many-to-many relationship between record occurrences of the same record type. The member record type is the junction record type between the two related owners.

For a simple bill-of-materials structure, the server requires that the owner record type be represented as two or more segments with different field names for the identical fields. This ensures that, at retrieval time, the field names specified in the user's request will provide the proper navigational path.

In the following graphic, the EMPLOYEE and STRUCTURE record types are connected by two sets. This simple bill-of-materials structure can be described two ways:

1. As an employee-to-manager relationship. The EMPLOYEE record type is the parent segment of the non-unique STRUCTURE segment using the REPORTS-TO set. The STRUCTURE record type, in turn, is the parent segment of the unique MANAGER segment using the MANAGES set (the MANAGER segment duplicates the EMPLOYEE segment and its fields are renamed).
2. As an employee-to-subordinate relationship. The EMPLOYEE record type is the parent segment of the non-unique STRUCTURE segment using the MANAGES set. The STRUCTURE record type is the parent segment of the unique SUBORD segment using the REPORTS-TO set (the SUBORD segment duplicates the EMPLOYEE segment and its fields are renamed).



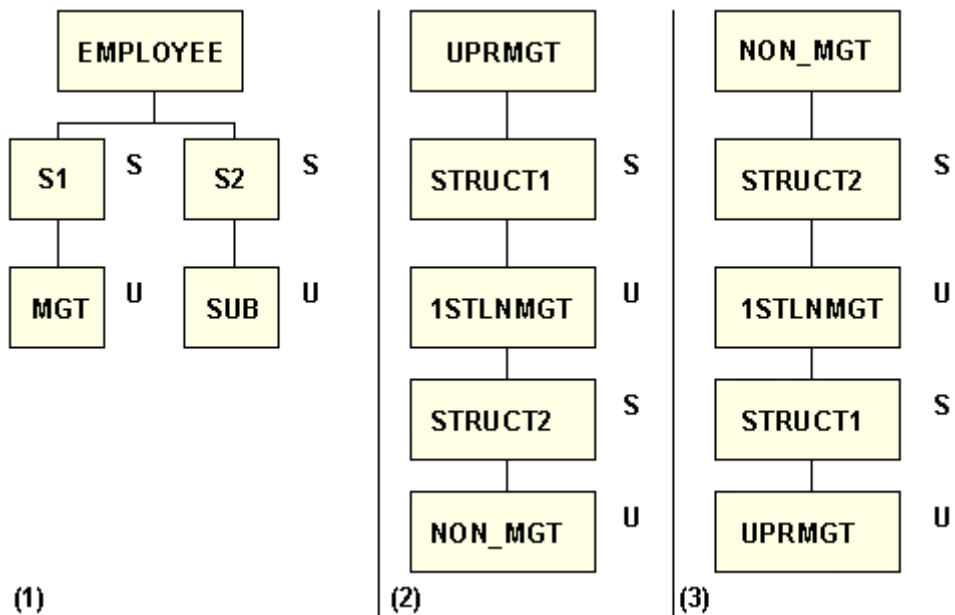
The previous graphic represents a two-tiered employee-to-employee relationship. Multi-tiered relationships are extended bill-of-materials structures. Multi-tiered relationships are created and used for different levels of answer sets. The number of levels or tiers should be kept to a minimum.

To determine the number of segments required to describe an n-tiered relationship, use this formula:

$$\text{Number of segments} = (2 \times \text{number of tiers}) - 1$$

The following graphic is a three-tiered version of the previous graphic:

1. The first panel combines both views with EMPLOYEE as the parent of two non-unique descendants, S1 and S2. Both S1 and S2, in turn, are parents of a unique descendant, MGT and SUB, respectively (S1 and S2 describe junction records that point to MGT and SUB).
2. The second panel shows a three-tiered relationship between employees implemented in a five-segment single-path hierarchy. The segments UPRMGT (upper management), 1STLNMGT (first-line management), and NON_MGT (non-management) all describe the EMPLOYEE segment but have renamed field (like S1 and S2 above, STRUCT1 and STRUCT2 contain renamed fields that point to descendant segments).
3. The third panel depicts the opposite of panel 2.

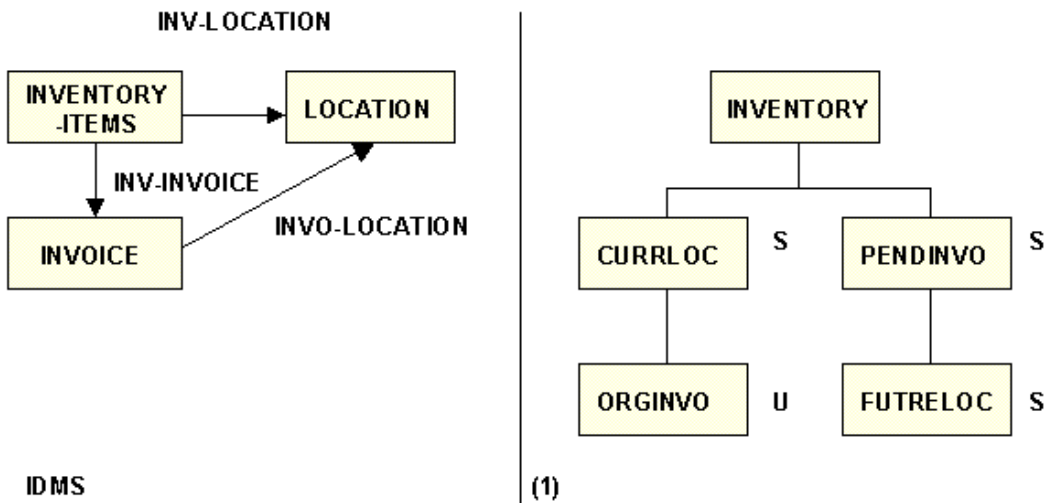


Loop Structures

Loop structures in IDMS/DB implement complicated relationships among record types. For the server depiction of a loop, you must select a record type to be the parent in the relationship.

Suppose you had a loop structure like the one below. An INVOICE record occurrence has an owner record occurrence (INVENTORY-ITEMS) only if the invoice order is pending or has not been delivered. The INVO-LOCATION set lists the location where an invoice item will be or was delivered.

In panel 1 in the following graphic, the server translates this structure as multi-tiered, with one tier of renamed segments. The INVENTORY-ITEMS record type, in this case, acts as the root INVENTORY. The LOCATION record type is mapped as the CURRLOC segment that lists the location of items currently in stock. The INVOICE record type is mapped as the PENDINVO segment that lists pending invoice orders. Segments ORGINVO and FUTRELOC are required segments that rename the data in record types INVOICE and LOCATION. The segment ORGINVO contains historical information for an inventory item. The segment FUTRELOC indicates where an ordered item will be delivered.



CALC-Based and Index-Based Relationships

Logical relationships, unlike physical ones, are based on the occurrence of the same data value in two different record types. To make the parent/descendant connection, the Data Adapter for IDMS/DB uses CALC fields or indices to locate the related record occurrence(s). The related fields are not required to have the same name in both record types, but the field format must be the same.

Note:

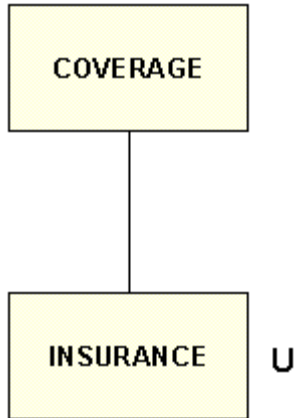
- WHEREs on CALC fields and indices are also used to generate CALC and index calls to IDMS/DB.
- CALC or index fields can also be GROUPNAMEs, consisting of fields contiguous in both parent and descendant segments. The format types and lengths of the GROUP and its fields must be comparable in both parent and descendant segments.

Like set-based descendants, CALC- and index-based descendants are unique or non-unique, but this depends largely on how the DUPLICATES parameter is specified in the Access File (the SEGTYPE parameter for the descendant must also reflect the DUPLICATES parameter; for example, CLCDUP=N, SEGTYPE=U). The server treats unique descendants in the same manner regardless of what underlies the parent/descendant relationship: set, index, or CALC field.

The COVERAGE and INSURANCE-PLAN record types in the graphic in *Set-Based Relationships* on page 10-5 illustrate a logical relationship. The field PLAN-CODE is common to both record types. The parent/descendant relationship can be described to the server if:

- The INSURANCE-PLAN record type has an index on PLAN-CODE.
- The INSURANCE-PLAN record type is an IDMS/DB CALC record type with PLAN-CODE as the CALC key.

The graphic below depicts the server representation of the graphic in *Set-Based Relationships* on page 10-5 that uses the CALC field method. The server interprets COVERAGE as the parent segment and INSURANCE-PLAN as the descendant. Since the DUPLICATES parameter is set to N (CLCDUP=N), the INSURANCE segment is unique.



Logical Record Facility Concepts

The Data Adapter for IDMS/DB supports two kinds of LRF-based records.

- Logical Records (LR) are built from network record types. They are not physically stored as such; instead, they are defined within a subschema using DBA-supplied navigational paths. Logical Records are created at execution time from real database records—types based on DBA-supplied navigational paths and the SELECTs passed to LRF from the server.
- Automatic System Facility (ASF) records are created by end-users with the Automatic System Facility, a menu-driven front end to LRF. This Facility generates subschemas and navigational paths based on the user's menu selection.

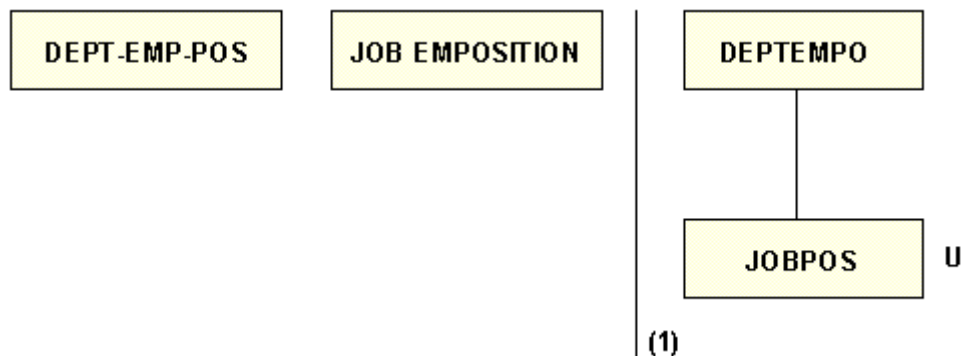
The Data Adapter for IDMS/DB uses the IDMS/DB Logical Record Facility (LRF) to retrieve and create ASF and LR records. In general, the LRF-based records can contain information from several sources—both DML-created and ASF-generated—that are located in several database areas. All fields, records, and areas, however, must be defined to the same subschema.

LRF-based records may be related to each other using an embedded cross-reference. Your Master File can list up to 64 related LRF-based records. Access Files can list an unlimited number of subschemas. However, the server accesses up to 16 subschemas per request.

LRF Records as Descendants

With the Data Adapter for IDMS/DB/DB, you may create a parent/descendant relationship between two LRF records if they share a common field or GROUP. Field lengths and formats must be the same in both segments. The shared fields are specified in the descendant segment declaration of the Access File as the values of the KEYFLD/IXFLD pair.

For example, in the graphic below, the LRF record DEPT-EMP-POS is represented as the root segment DEPTEMPO; the LRF record JOB-EMPOSITION as the unique segment JOBPOS. The related fields are POS_STRT_DR2 (the IXFLD value) and POS_STRT_DT1 (the KEYFLD value).



The restriction that the shared field in the descendant record be a CALC or an indexed field does not apply to LRF records. However, this kind of embedded cross-reference has one restriction: The IDMS/DB path group for an LRF record that acts as the descendant segment must contain a SELECT clause to process the implied WHERE clauses.

Syntax

How to Implement a Parent/Descendant Relationship With SELECT

SELECT clauses are maintained by your DBA and should already be available in your subschema. The following SELECT clauses can successfully implement a parent/child relationship. They are listed in descending order of efficiency:

1. `SELECT FOR FIELDNAME EQ fieldname`
2. `SELECT USING INDEX indexname`
3. `SELECT FOR FIELDNAME fieldname`
4. `SELECT FOR ELEMENT elementname`
5. `SELECT`

where:

fieldname

Is the joined-to IDMS/DB field name on the descendant record type.

indexname

Is the index set name to the joined-to field.

elementname

Is the physical record type on which the joined-to field is stored.

Note:

- If no specific SELECT clause exists, a null SELECT clause (item 5) must be available to process an answer set.
- LRF records that return partial record occurrences and user-defined path status should not be used. If the adapter encounters a user-defined path status, the status is interpreted as LR-ERROR and the retrieval process is halted with error messages EDA967 and EDA949.
- Do not confuse the use of the word SELECT with the SQL SELECT statement. In the above example, the word SELECT is used to refer to the IDMS/DB internal meaning.

Summary of Network Relationships

As illustrated in the previous examples, there are four ways to implement a network relationship. Each can be described as a parent/descendant relationship:

- Owner/member
- Member/owner
- Field or GROUP/CALC field
- Field, GROUP/SPF, or Integrated Index

All four can be intermixed in one Master File. The actual underlying connection (set, CALC, index) between parent and descendant is not apparent to end users who specify field names. In addition, a Master File can list record types from multiple subschemas, databases, and dictionaries.

In server processing, the identity of the record type behind a segment is invisible. The retrieval technique is followed in all cases as if all segments were represented by distinct records. Within IDMS/DB, currencies are maintained by storing the DBKEYs of record types and retrieving the record occurrence again, when needed.

The top-to-bottom order in which segments are defined (the chains of parent/descendant relationships) corresponds to structural relationships among the IDMS/DB record types. This top-to-bottom order is logically significant; the left-to-right order, on the other hand, is not.

Managing IDMS/DB Metadata

This section explains the rules for making an IDMS/DB data source accessible to the server using the Master File. The Access File is an extension of Master File syntax that provides information to map DML record types and LRF-based records, including record and area names and, where needed, set or field name information.

Note: This section relates to the describing of IDMS/DB data sources for retrieval in the SQL language.

Master Files for IDMS/DB

A Master File consists of file, segment, and field declarations. Rules for declarations are:

- Each declaration must begin on a separate line and be terminated by a comma and dollar sign (,\$). Text appearing after the comma and dollar sign is treated as a comment.
- A declaration can span as many lines as necessary as long as no attribute=value pair is separated. Commas separate attribute=value pairs.

The following sections summarize the syntax for each type of declaration, and then describe each attribute.

File Attributes

Master Files begin with a file declaration that names the file and describes the type of data source; in this case, an IDMS/DB data source. The file declaration has two attributes, FILENAME and SUFFIX.

Syntax

How to Identify a Master File

```
FILE[NAME]=name, SUFFIX=IDMSR [, $]
```

where:

name

Is any 1 to 8 character name. On OS/390 and z/OS, the Master File is the member name within the PDS allocated to DDNAME MASTER.

IDMSR

Indicates that the Data Adapter for IDMS/DB is required for data retrieval.

Segment Attributes

Each IDMS/DB segment described in a Master File requires a segment declaration that consists of at least two attributes, SEGNAME and SEGTYPE.

- SEGNAME
- SEGTYPE
- PARENT
- CRFILE (Cross-referenced File)
- OCCURS and POSITION

SEGNAME

The SEGNAME value assigned to DML record types and LRF records are suggestive names -- not necessarily the IDMS/DB record names. The segment name may be a maximum of eight characters and must be unique within a given Master File. If the same IDMS/DB record type is viewed in two different contexts (for example, a loop structure), two segment declarations are required.

SEGTYPE

The SEGTYPE keyword indicates whether a segment occurs once (SEGTYPE=U) or many times (SEGTYPE=S). It is used as follows:

- For a root segment, SEGTYPE has no meaning and may be omitted entirely.
- For a descendant segment, SEGTYPE values can be S or U.
- For descendant segments with set-based relationships, SEGTYPE indicates whether the segment acts as an owner or a member. If the descendant segment is the owner record type, the SEGTYPE value is U. If the descendant is the member, the SEGTYPE value is S.
- For descendant segments with CALC-based or index-based relationships, SEGTYPE values may be S or U depending on the DUPLICATES (CLCDUP or IXDUP) value.
- For descendant segments with LRF-based relationships, SEGTYPE values may be S or U.

PARENT

The PARENT keyword is required for descendant segment declarations. The PARENT value names the descendant's parent segment. This keyword is not specified for the root declaration.

CRFILE (Cross-referenced File)

The CRFILE keyword is specified only for segments that are described remotely in another Master File. The field descriptions for these segments are, in effect, copied into the Master File at execution time. Remote descriptions are discussed in *Remote Descriptions* on page 10-25.

OCCURS and POSITION

The OCCURS and POSITION attributes are specified only when the segment corresponds to an intra-record structure, such as a COBOL OCCURS or OCCURS DEPENDING clause. These keywords are described in detail in *Intra-record Structures: OCCURS Segment* on page 10-26.

Field Attributes

Each segment consists of one or more fields. The Master File need not describe all fields from a segment. The attributes are:

- FIELDNAME
- ALIAS
- USAGE
- ACTUAL
- GROUP Fields
- IDMS/DB Database Key

To describe a field in the Master File, you must specify the primary attributes FIELDNAME, ALIAS, USAGE, and ACTUAL. These attributes are discussed in this section. Note that the adapter does not support the MISSING attribute.

FIELDNAME

The FIELDNAME keyword may contain any name -- not necessarily an IDMS/DB field name. It can be a maximum of 48 characters. The name that you assign must be unique, because in the server, data is referenced through field names. If the same IDMS/DB record type viewed in different contexts underlies two or more segments, different field names must be specified in separate segment descriptions.

Due to the record-oriented nature of IDMS/DB, the fields are described for each segment in the same order as they appear in the subschema record area. Bytes skipped for field alignment or for fields you want to omit must be described as dummy fields of the appropriate length.

For example, a 1-byte alphanumeric field is followed by a double-precision floating point field:

```
SEGNAME=EMPSTATUS,$
  FIELD=STATUS_CODE,SCODE,A1,A1,$
  FIELD=                ,                ,A7,A7,$
  FIELD=GROSS_SALES,GSales,D12.2,D8,$
```

The same name can be used for all dummy fields; in the example above, blanks are used.

You do not need to describe all fields of the record type or LRF record; only an initial set, starting with the first field and continuing up to the last field of interest. For variable-length record types, treat the fixed-length portion as one segment and the variable portion as another segment, as described with the OCCURS attribute. For information about OCCURS segments, see *Intra-record Structures: OCCURS Segment* on page 10-26.

Syntax

How to Describe a Field in the Master File

```
FIELD[NAME]=field,[ALIAS=]alias,[USAGE=]display,[ACTUAL=]format,$
```

where:

field

Is a 1 to 48 character field name.

alias

The ALIAS keyword is used for certain fields which require specific ALIAS values.

display

Is the display format for the field.

format

Is the definition of the IDMS/DB field format and length (n).

You can omit the ALIAS, USAGE, and ACTUAL keywords from the field declaration if the values are specified in the standard order (FIELD, ALIAS, USAGE, ACTUAL). For example, the following declarations are equivalent:

```
FIELD = YEAR, ALIAS=, USAGE=A2, ACTUAL=A2,$
FIELD = YEAR, ,A2, A2,$
```

ALIAS

The ALIAS keyword is used for certain fields which require specific ALIAS values:

- Database key fields -- the ALIAS value is DBKEY.
- ORDER fields -- the ALIAS value is ORDER.
- Fields on LRF-based records (LR or ASF) that correspond to the last fields of their underlying physical record types. Filler fields may be required if the last field does not end on a double-word boundary. The ALIAS for each filler field is a unique name with a maximum of eight characters counting the .END suffix. The adapter uses this suffix to correctly address LRF records for LRF calls.
- GROUP fields -- an ALIAS is always required or an error message results.

The following is an example of a filler field in a segment that describes an LRF record. Suppose a record called JOB-EMPOSITION is defined to the IDMS/DB subschema with seven fields: the first three fields are derived from a physical (DML) record type called JOB; the last four fields, from the EMPOSITION physical record type. The Master File syntax for this LRF record looks like this:

```
FILENAME=JOBMAST , SUFFIX=IDMSR , $
  SEGNAME=JOBEMP , SEGTYPE=S , $
    FIELDNAME=JOBID      , ALIAS=          , USAGE=A4      , ACTUAL=A4      , $
    FIELDNAME=TITLE      , ALIAS=          , USAGE=A20     , ACTUAL=A20     , $
    FIELDNAME=DESCRIPTN  , ALIAS=CMTS.END , USAGE=A120    , ACTUAL=A120    , $
    FIELDNAME=START_DTE  , ALIAS=          , USAGE=A6YMD   , ACTUAL=A6      , $
    FIELDNAME=FINISH_DTE , ALIAS=          , USAGE=A6YMD   , ACTUAL=A6      , $
    FIELDNAME=SALARY_GRADE , ALIAS=          , USAGE=P4      , ACTUAL=Z2      , $
    FIELDNAME=SALARY     , ALIAS=          , USAGE=P10.2  , ACTUAL=P5      , $
    FIELDNAME=           , ALIAS=FILL.END , USAGE=A1      , ACTUAL=A8      , $
```

In this example, the filler field corresponds to the last field of the EMPOSITION record type, because LRF records must lie on a double-word boundary. The IDMS/DB filler field is stored to make the record length a multiple of 8. IDMS/DB filler fields on physical record types underlying LRF views must become filler fields. These filler fields can have any field name or contain a blank; its ALIAS must include the .END suffix. Since the JOB record type is 144 bytes (total of ACTUAL formats), it does not need a filler; the last field DESCRIPTN only requires an alias with an .END suffix.

USAGE

Master Files require you to specify field formats and lengths for USAGE and ACTUAL parameters.

The USAGE and ACTUAL keywords describe the data format of the field. The USAGE parameter defines the field length for the fields. Allow for the maximum possible number of characters or digits including decimal points. You may include valid edit options without increasing the length size.

Note: For network record types, the ACTUAL and USAGE formats of zoned CALC and zoned index fields must be described as alphanumeric (A).

ACTUAL

The ACTUAL parameter defines the field length for COBOL fields found in the IDMS/DB file. The number of internal storage bytes used by COBOL determines the field's actual length for these formats:

Alphanumeric (A)	Equals the number of characters described in the PICTURE clause.
Zoned decimal (Z)	Equals the number of characters described in the PICTURE clause.
Integer (I)	Equals 2 or 4, corresponding to decimal lengths of 1-4 or 5-9 in the PICTURE clause.
Floating-point (F)	Equals 4 bytes.
Double-precision (D)	Equals 8 bytes.
Packed decimal (P)	Equals (number of PICTURE digits / 2) + 1; excluding sign (S) or implied decimal (V).

Use the following chart as a guide for describing ACTUAL formats:

COBOL Format	COBOL PICTURE	Bytes of Storage	ACTUAL Format	USAGE Format
DISPLAY	X(4)	4	A4	A4
DISPLAY	S99	2	Z2	P3
DISPLAY	9(5)V9	6	Z6.1	P8.1
DISPLAY	99	2	A2	A2

COBOL Format	COBOL PICTURE	Bytes of Storage	ACTUAL Format	USAGE Format
COMP	S9	4	I2	I1
COMP	S9(4)	4	I2	I4
COMP	S9(5)	4	I4	I5
COMP	S9(9)	4	I4	I9
COMP-1	-	4	F4	F6
COMP-2	-	8	D8	D15
COMP-3	9	1	P1	P1
COMP-3	S9V99	2	P2	P5.2
COMP-3	9(4)V9(3)	4	P4	P8.3
FIXED BINARY(7) (COMP-4)	B or XL1	4	I4	I7

Note:

- For COMP-1 and COMP-2, allow for the maximum possible digits.
- For COBOL DISPLAY fields with zoned decimal, server formats must be packed (P).
- For COMP-1 and COMP-2, PICTURE clauses are not permitted for internal floating-point formats (F and D).

GROUP Fields

The GROUP keyword identifies a set of fields following it with a single name. This GROUP name is any unique name up to 48 characters in length. Its usage is similar to that of a COBOL group name. Generally, this attribute is used for IDMS/DB indexed or CALC fields.

Note:

- USAGE and ACTUAL format types for a GROUP field are always alphanumeric (A).
- USAGE and ACTUAL format lengths for a GROUP field are the sums of the field lengths that form the GROUP field.

For example, a GROUP field called EMP_NAME is composed of two fields, FIRST_NAME and LAST_NAME. Notice the USAGE and ACTUAL formats.

```
GROUP=EMP_NAME      ,ALIAS=      ,USAGE=A25  ,ACTUAL=A25  ,$
  FIELD=FIRST_NAME ,ALIAS=      ,USAGE=A10  ,ACTUAL=A10  ,$
  FIELD=LAST_NAME  ,ALIAS=      ,USAGE=A15  ,ACTUAL=A15  ,$
```

The FIELDTYPE keyword identifies indexed fields, both SPF and Integrated, on network record types. Omit it for LRF records.

IDMS/DB Database Key

The database key of a network record type corresponding to a segment can optionally be described as the last field in the segment. To specify a database key, use the following values:

Keyword	Description
FIELDNAME	Is any unique key. It can be a maximum of 48 characters.
ALIAS	Value is DBKEY.
USAGE	Value is I10.
ACTUAL	Value is I4.

The database key is used to select records from entry segments when screening on particular IDMS/DB values in the user request.

Remote Descriptions

SEGTYPEs KLU and KL specify unique and non-unique segments for which the aggregate fields are remotely described in another Master File. In other words, a KLU or KL segment contains no field information; its fields are fully defined in another Master File. The CRFILE keyword specifies the source or remote Master File. At execution time, the remote field descriptions are, in effect, copied into the current Master File.

Generally, KLU and KL segment types are used when several Master Files are constructed for the same IDMS/DB subschema. They save typing and maintenance effort. Use KLU if the underlying segment is unique (U); KL if the underlying segment is non-unique (S). Remote descriptions have no logical implications regarding parent/descendant relationships, nor do they impact their implementation.

To specify a remote description, you must give the KLU or KL segment the same name as the source segment in the remote Master File to ensure the proper use of field descriptions. The source file is specified as the CRFILE value. For example:

```
SEGMENT=SALES , PARENT=DEPT , SEGTYPE=KL , CRFILE=RECORDS , $
```

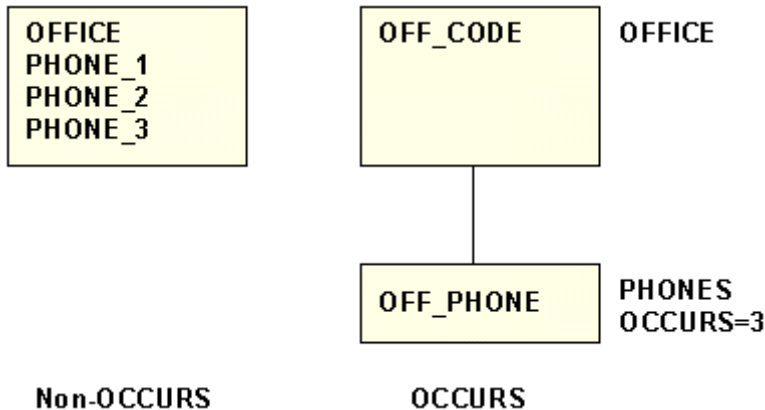
This segment declaration indicates that the field descriptions for the SALES segment are obtained from the SALES segment in the RECORDS Master File. The SEGTYPE for the SALES segment in the RECORDS Master File cannot be KL or KLU—only U or S. Only field names and their attributes from the source file are used; original segment attributes are not. The source file does not have to be a file description as long as it describes the named segment.

Intra-record Structures: OCCURS Segment

This section describes the OCCURS segments.

A common record structure is one where a field or group of adjacent fields is repeated in the same record type. In COBOL and PL/I syntax, repeating intra-record structures are defined through an OCCURS clause. The server equivalent of an OCCURS clause is an OCCURS segment.

For example, the OFFICE record type contains the field OFFICE-PHONE that occurs three times. The corresponding OFFICE segment can list three separate fields with different field names, or alternately OFFICE-PHONE can be described in a separate descendant OCCURS segment. Using the OCCURS method, each office phone is referenced by a single name (see the Master File for EMPSS01). The graphic below depicts the non-OCCURS method and the OCCURS method.



OCCURS segments have two attributes or keywords: ORDER and POSITION. The POSITION keyword directs the server to OCCURS segments when non-repeating fields exist between repeating fields. The ORDER keyword creates a fictitious count field that may be decoded.

Describing the Repeating Group to the Server

Any fixed or variable-length record type described in COBOL can be mapped into a server hierarchy using OCCURS segments. A simple OCCURS segment is a descendant of the parent segment which contains non-repeating fields found in the IDMS/DB record type. You must specify the OCCURS keyword on the descendant segment declaration that describes the repeating group. Like the COBOL OCCURS clause, the value of this keyword may be a numeric constant or a field name. The numeric constant indicates a fixed number of repetitions; a field name indicates a count field in the parent segment that maintains a count of the number of occurrences.

OCCURS segments also describe parallel and nested intra-record hierarchical structures. Parallel sets of repeating groups are described as multiple descendant segments of the same parent. In a nested structure, where a repeating group contains another repeating group, one OCCURS segment is the parent of another. Fixed and variable OCCURS segments can be intermixed in any order.

The restrictions for OCCURS segments are as follows:

- The count field for a variably-occurring repeating group must be located physically before the repeating group in the parent of the OCCURS segment.
- A record structure that has a variable number of occurrences but no count field is not within the scope of IDMS/DB.
- The SEGTYPE for an OCCURS segment is specified as S or KL, indicating that it is a non-unique segment.
- OCCURS segments must be defined in the Master File in the same order as they appear in the actual record type, unless the POSITION attribute is used.
- OCCURS segments do not have a corresponding segment declaration in the Access File, because the Data Adapter for IDMS/DB simulates them using the parent record. OCCURS segments do not generate any additional IDMS/DB calls.

From the server standpoint, OCCURS segments are indistinguishable from other segments. The server processes them, if referenced, in the usual top-to-bottom left-to-right retrieval order.

For example, consider this COBOL budget record type:

```
01 BUDGET-RCD.
02 ACCOUNT          PIC XXX.
02 ACTUAL-COUNT     PIC 99.
02 PLANNED-AMT      PIC 9(9) OCCURS 12 TIMES.
02 ACTUAL-AMT       PIC 9(9) OCCURS 12 TIMES
                    DEPENDING ON ACTUAL-COUNT.
```

The equivalent Master File is:

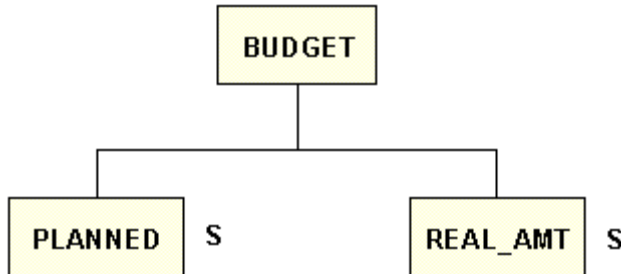
```
SEGMENT=BUDGET , SEGTYPE=S , $
  FIELD=ACCOUNT      , ALIAS=      , USAGE=A3   , ACTUAL=A3   , $
  FIELD=ACTUAL_COUNT , ALIAS=      , USAGE=P4   , ACTUAL=Z2   , $

SEGMENT=PLANNED , PARENT=BUDGET , SEGTYPE=S , OCCURS=12 , $
  FIELD=PLANNED_AMT , ALIAS=      , USAGE=P12  , ACTUAL=Z9   , $

SEGMENT=REAL_AMT , PARENT=BUDGET , SEGTYPE=S , OCCURS=ACTUAL_COUNT , $
  FIELD=ACTUAL_AMT  , ALIAS=      , USAGE=P12  , ACTUAL=Z9   , $
```

The mapping principle is very simple: the non-repeating field in the record type is described in one parent segment, and the parallel COBOL OCCURS structures are made into descendant OCCURS segments. The OCCURS keyword on the descendant segments specifies either a number (fixed occurrences) or a count field (variable occurrences). In this case, the count field ACTUAL_COUNT is located in the immediate parent segment called BUDGET and is specified on the REAL_AMT descendant segment.

The following is a diagram for this Master File example:



If the PLANNED or REAL_AMT segments had repeating structures, they would in turn be parents of OCCURS segments defined by the same principles. The BUDGET segment could have other non-OCCURS descendants. The PLANNED and REAL_AMT segments might have CALC- or index-based descendants. However, set-based descendants of this record type would be tied to the BUDGET segment -- not to the PLANNED or REAL_AMT segments.

POSITION

As noted in the previous section, OCCURS segments are defined in the same order as they appear in the actual record type. In some cases, COBOL OCCURS clauses are separated by non-repeating fields. The POSITION keyword in a Master File indicates that the repeating fields are located in the middle of non-repeating fields.

The POSITION attribute can only be used for a repeating group with a fixed number of occurrences. This means that the OCCURS attribute of the descendant segment must equal a numeric constant and not a count field.

Suppose the previous COBOL record type looked like this:

```
01 BUDGET-RCD.  
02 ACCOUNT          PIC XXX.  
02 PLANNED-AMT     PIC 9(9) OCCURS 12 TIMES.  
02 ACTUAL-COUNT    PIC 99.  
02 ACTUAL-AMT     PIC 9(9) OCCURS 12 TIMES  
                   DEPENDING ON ACTUAL-COUNT.
```

Here, the two repeating structures PLANNED-AMT and ACTUAL-AMT are separated by the non-repeating field ACTUAL-COUNT, which clearly belongs to the BUDGET segment. You must indicate in the Master File that the first occurrence of the PLANNED segment will not immediately follow the ACCOUNT field in the BUDGET segment (the PLANNED-AMT field is described in a separate descendant segment). The POSITION keyword accomplishes this task by directing the server to the descendant segment named PLANNED.

The corresponding Master File looks like this:

```
SEGMENT=BUDGET , SEGTYPE=S , $
  FIELD=ACCOUNT      , ALIAS=      , USAGE=A3      , ACTUAL=A3      , $
  FIELD=PLANNED_SEG1 , ALIAS=      , USAGE=A108    , ACTUAL=A108    , $
  FIELD=ACTUAL_COUNT , ALIAS=      , USAGE=P4      , ACTUAL=Z2      , $

SEGMENT=PLANNED  , PARENT=BUDGET , SEGTYPE=S , OCCURS=12 ,
  POSITION=PLANNED_SEG1      , $
  FIELD=PLANNED_AMT  , ALIAS=      , USAGE=P12    , ACTUAL=Z9      , $

SEGMENT=REAL_AMT , PARENT=BUDGET , SEGTYPE=S , OCCURS=ACTUAL_COUNT , $
  FIELD=ACTUAL_AMT  , ALIAS=      , USAGE=P12    , ACTUAL=Z9      , $
```

The POSITION keyword in the PLANNED segment names a field called PLANNED_SEG1 in BUDGET, the immediate parent segment. PLANNED_SEG1 in the parent coincides with the first field of the first occurrence in PLANNED, the OCCURS segment. The REAL_AMT segment does not require a POSITION keyword, because the position of its first occurrence is correctly inferred as following the last described field in the BUDGET segment.

In the previous example, the PLANNED_SEG1 spans all occurrences of the PLANNED segment. As an alternative, 12 individual fields named PLANNED_SEG1 through PLANNED_SEG12 could be described instead in the BUDGET segment. Each individual field would need the appropriate numeric format. Then you could refer to any one of the 12 amount fields by its specific name, or generically through the PLANNED_AMT field. The POSITION attribute can always be used for this purpose, even when it is not required for positioning the first position of an OCCURS segment located in the middle of the record type.

ORDER Field

Sometimes the sequence of fields within an OCCURS segment is significant. For example, each instance of the repeating field may represent one quarter of the year, but the segment may not have a field that specifies the quarter to which it applies.

ORDER is an optional counter used to identify the sequence number within a group of repeating fields. Specify it when the order of data is important. The ORDER field does not represent an existing field in the data source; it is used only for internal processing.

Syntax **How to Identify an ORDER Field**

The ORDER field must be the last field described in the OCCURS segment.

```
FIELDNAME=name , ALIAS=ORDER, USAGE=In, ACTUAL=I4 , $
```

where:

name

Is any valid field name.

In

Is an integer format.

Note:

- The ALIAS value must be ORDER.
- The ACTUAL format must be I4.
- The ORDER field must be the last field defined in the OCCURS segment.

In requests, you can use the value of the ORDER field. You can also specify a DEFINE statement in the Master File to translate it to more meaningful values. For example:

```
DEFINE QTR/A3 = DECODE ORDER(1 '1ST' 2 '2ND' 3 '3RD' 4 '4TH');
```

A subsequent request could include

```
SELECT TOT.TAXES FROM JOBMAST  
WHERE QTR = 1
```

or

```
SELECT QTR, BALANCE, INTEREST
```

Access Files for IDMS/DB

An Access File describes the database access information; for example, database number, file number, and descriptor fields.

Note that before you can access a CA-IDMS/DB file, you must first describe it to the server in two files: a Master File, and an associated Access File. A Master File is required to describe the CA-IDMS/DB file in standard server terminology to the Data Adapter for IDMS/DB/DB. For more information on the Master File, see *Managing IDMS/DB Metadata* on page 10-17.

An Access File is required to translate a request for network record types and LRF records into the appropriate CA-IDMS/DB Data Management Language (DML) retrieval commands. This file description consists of 80-character records called declarations in comma-delimited format (keyword=value).

An Access File contains three kinds of declarations:

- Subschema
- Segment
- Index

Subschema declarations identify the subschema used, the IDMS/DB release under which the subschema was compiled, the calling mode to be used to retrieve records, and whether a trace is to be produced. Several subschema declarations may be specified in a single Access File. Each subschema declaration is followed by its segment and index declarations. *Subschema Declaration Keywords* on page 10-32 describes keywords for subschema declarations.

Access segment declarations indicate the IDMS/DB record information and the parent/descendant relationship for each network record type or LRF record described as a segment in a Master File (access segment declarations are not defined for OCCURS segments). Segment declarations can be specified in any order after their corresponding subschema declaration. *Segment Declaration Keywords for Network Record Types* on page 10-34 and *Segment Declaration Keywords for LRF Records* on page 10-37 describe segment declarations for network record types and LRF records, respectively.

Index declarations provide information about each IDMS/DB index. They may also be in any order—one for each indexed field described in the Master File—after their corresponding subschema declaration. *Index Declaration Keywords for Network Record Types* on page 10-39 describes index declarations for network record types.

CA-IDMS/DB Access File Syntax

Each declaration consists of a list of keyword and value pairs, separated by commas. The list is free-form and may span several lines; keywords may be specified in any order. Each declaration is completed with a comma followed by a dollar sign (\$). For example, this Access File contains three declarations:

```
SSHEMA=PAYROLL, RELEASE=15, INDEXAREA=PRIMARY-IX-AREA,
    DBNAME=EMPDEMO, DICTNAME=APPLDICT, $
```

```
SEGNAME=ACCOUNT, RECORD=PAYREC, AREA=PAY-REGION,
    CLCFLD=EMPLOYEE_ID, CLCDUP=N, $
```

```
IXSET=IXREC-SSN, IXAREA=IX-AREA1, IXFLD=PERS_SSN,
    IXDUP=N, IXORD=A, $
```

Blank lines and lines starting with an asterisk (*) in column 1 are treated as comments. Leading and trailing blanks around keywords and values are ignored. Values that contain commas, equal signs, dollar signs, or spaces must be enclosed in single quotation marks.

Subschema Declaration Keywords

If your Master File defines record types and LRF records from multiple IDMS/DB subschemas, the Access File should contain multiple subschema declarations. After each subschema declaration, list its segment and index declarations.

Subschema declarations for DML and LRF subschemas contain the following keywords; certain keywords are optional as explained in the following summary chart.

Keyword	Description
SSHEMA	Is the IDMS/DB subschema name.
RELEASE	Is the release of the IDMS/DB software that was used in the last compilation of the subschema. The value can be 10.2, 12, or 12.x (where x is your release version), 14, or 15.
MODE	Indicates the type of IDMS/DB access the adapter is to perform. Specify LR for LR and ASF records; DML is the default.
TRACE	Is an optional keyword used for debugging purposes. It specifies whether a basic trace of all IDMS/DB calls or a detailed trace of all the parameters passed to IDMS/DB will be displayed. Values can be YES, PARMS, or NO (NO is the default).

Keyword	Description
READY	Is an optional keyword that specifies when an LRF record is built from two or more physical record types located in several database areas. The adapter prepares or readies all the areas of the subschema. Values can be ALL or null or omitted entirely.
DBNAME	Specifies the IDMS/DB database name from the DBNAME table corresponding to its subschema. It can be used in local or Central Version mode to translate the subschema name into the proper load module(s) for data access.
INDEXAREA	Is the name of the primary SPF index area. Required for any subschema with SPF indices.

CV Mode Only

Keyword	Description
DICTNAME	Identifies a secondary dictionary load area, if the subschema is not located in the primary dictionary or in a load PDS. Remember that ASF subschemas are located in secondary dictionary load areas by default; so if your Access File describes an ASF record, you must specify this keyword.
NODE	Identifies the Distributed Database Systems (DDS) node location of an IDMS/DB distributed database. The value is the IDMS/DB data dictionary table entry that identifies the DDS node location of an IDMS/DB distributed database. This keyword is required only if DDS is installed at a user site and if the subschema is located in a remote site location.
DICTNODE	Identifies the DDS node location of an IDMS/DB distributed database subschema in a secondary dictionary load area. The value is the IDMS/DB data dictionary table entry that identifies the DDS node location of an IDMS/DB distributed database subschema. This keyword is required only if DDS is installed at the user site and if the subschema is located in a remote site location.

Note: When running using DDS, Central Version must be used. DDS access is not supported in local mode.

Segment Declaration Keywords for Network Record Types

Your use of keywords in segment declarations for DML record types depends on whether the record type contains a CALC key, acts as a descendant segment, or contains an index. The following keywords are common to all segment declarations.

Keyword	Description
SEGNAM	Is the corresponding Master File segment name of the DML record type.
RECORD	Is the IDMS/DB record type name.
AREA	Is the IDMS/DB area name that contains the record type.

If your record type is CALC (that is, if the record contains a CALC key) include the two keywords below. These keywords are required for all CALC record types, regardless of how their parent/descendant relationships are implemented.

Keyword	Description
CLCFLD	Is the Master File field name of the CALC field.
CLCDUP	Indicates if the CALC field allows duplicates. The value can be Y or N.

Record types are assigned parent/descendant relationships in the server. These relationships are based on sets and CALC or index fields. Keywords for descendant segments follow. Consult your Master File to determine whether a segment is a descendant or parent.

Field	Keyword	Description
	ACCESS	Indicates the relationship that exists between record types. The value CLC or IX specifies an embedded cross-reference based on a CALC or indexed field. The value SET indicates a physical relationship based on a set of pointers.
Set-Based (ACCESS=SET)	SETNAME	Is the name of the set relating a descendant to its parent.

Field	Keyword	Description
	SETMBR	Specifies whether the set membership is mandatory/automatic, mandatory/manual, optional/automatic, or optional/manual. This information is used to verify set membership at execution time. To determine the appropriate membership value, check the set label on your Bachman diagram. Values can be MA, MM, OA, or OM.
	GETOWN	Allows or inhibits GET OWNER calls which obtain the owner records from a member record type. If the value is Y, the adapter issues GET OWNER calls to retrieve the owner record in the set when SEGTYPE is U, KLU. If the value is N, GET OWNER calls are inhibited. Specify N only when the set has no owner pointers and long member record chains are apt to occur. When GET OWNER calls are inhibited, the owner record type cannot be a descendant of its member. In other words, if GET OWNER calls are inhibited, SEGTYPE cannot be U or KLU.
	MULTMBR	Indicates whether the set, in which this record type participates, contains more than one member record type. Values can be Y or N.
	KEYFLD	Is the server field that sequences the set. Select a field from a parent or descendant segment as the value of KEYFLD in the descendant segment declaration.
	SETORD	A (ascending) or D (descending) for sorted set sequence. This keyword is required.
	SETDUP	Y or N if duplicates are allowed. Required for sorted sets.

Field	Keyword	Description
CALC-Based (ACCESS=CLC)	KEYFLD	Provides the search value to read CALC and index fields in descendant segments. These search values are located in parent segment fields. Specify the parent field name for the value of KEYFLD in the descendant segment declaration. The KEYFLD keyword is especially important when the two record types in a parent/descendant relationship are from different subschemas. The record type that acts as the descendant segment is the entry point into the second subschema. It must have a CALC key (CLCFLD) or index set (SETNAME) with ACCESS=CLC or ACCESS=IX, respectively. The descendant segment declaration must also list the KEYFLD value from the parent segment in the first subschema.
Index-Based (ACCESS=IX)	KEYFLD	See above.
	SETNAME	Is the IDMS/DB name of the index set. A corresponding index declaration is required.

Example Describing One Subschema With Two Segments

This example shows one subschema with two segments that are CALC record types; the INVOICE record type has a set-based relationship with the CUSTOMER record type.

```
SSHEMA=SAMPSSCH, RELEASE=15,
SEGNAM=CUSTOMER, RECORD=CUSTOMER, AREA=CUSTOMER-REGION,
CLCFLD=CUST_NUMBER, CLCDUP=N, $
SEGNAM=INVOICE, RECORD=INVOICE, AREA=INVO-REGION,
CLCFLD=INV_NUMBER, CLCDUP=N, ACCESS=SET,
SETNAME=CUSTOMER-INVO, SETMBR=MA, GETOWN=Y, MULTMBR=N, $
```

Example Describing Two Subschemas

The following example shows two subschemas. The INSURANCE-PLAN record type has a CALC-based relationship with the COVERAGE record type.

```
SSHEMA=EMPSS01,RELEASE=15,
SEGNAM=COVERAGE,RECORD=COVERAGE,AREA=INS-DEMO-REGION,$
SSHEMA=EMPSS03,RELEASE=15,$
SEGNAM=INSURNCE,RECORD=INSURANCE-PLAN,
AREA=INS-DEMO-REGION,CLCFLD=INS_PLAN_CODE,CLCDUP=N,
ACCESS=CLC,KEYFLD=COV_CODE,$
```

If your record type contains an indexed field, you may suppress area sweeps when the segment is used as a point of entry into the database. To prevent area sweeps, specify the optional keyword below on the segment declaration. Only those record instances connected to the specified index field are accessed.

Keyword	Description
SEQFIELD	Is the Master File field name (FIELDTYPE=I) of the index.

Note: This optional keyword requires an index declaration (see *Index Declaration Keywords for Network Record Types* on page 10-39).

Segment Declaration Keywords for LRF Records

Segment declaration keywords for LR and ASF records are basically the same as those for network (DML) record types. Specified values, of course, differ.

Keyword	Description
SEGNAM	Is the corresponding Master File segment name of the LRF record.
RECORD	Is the IDMS/DB name of the LRF record.
LR	Value is Y.
AREA	Is the IDMS/DB area name that contains physical record types. Specify READY=ALL on the subschema declaration for more than one area (if fields originate from many areas).

LRF records use embedded cross-references to create parent/descendant relationships. Specify the following keywords for LRF records that act as descendant segments:

Keyword	Description
ACCESS	Value is LR.
KEYFLD	Is the Master File field name found in the parent.
IXFLD	Is the Master File field name found in the descendant.

The KEYFLD and IXFLD keywords are required to implement parent/descendant relationships. The KEYFLD keyword specifies a field from the parent segment that provides search values. The value of IXFLD, in turn, is a field in the descendant segment that contains equivalent values for KEYFLD. Any field may be selected for IXFLD provided that the record possesses a null SELECT clause.

Suppose your Master File for one subschema contained two LRF records that act as parent and descendant segments.

```
FILE=DEPTEMP , SUFFIX=IDMSR , $
SEGNAME=DEPTEMP , $
.
.
.
FIELD=EMP_ID , ALIAS= , USAGE=A4 , ACTUAL=A4 , $
SEGNAME=EMPJOB , PARENT=DEPTEMP , SEGTYPE=S , $
FIELD=EMPLOYEE_ID , ALIAS= , USAGE=A4 , ACTUAL=A4 , $
.
.
.
```

The common field is EMPLOYEE_ID; its field format is the same in both segments. The EMPJOB segment is the descendant of DEPTEMP. The descendant segment declaration in the Access File below contains KEYFLD and IXFLD values.

```
SSHEMA=EMPSS06 , RELEASE=15 , MODE=LR , READY=ALL
DBNAME=EMPDEMO , DICTNAME=APPLDICT , $
SEGNAM=DEPTEMP , RECORD=DEPT-EMPLOYEE ,
AREA=EMP-DEMO-REGION , LR=Y , $
SEGNAM=EMPJOB , RECORD=EMPLOYEE-JOB ,
AREA=ORG-DEMO-REGION , LR=Y ,
ACCESS=LR , KEYFLD=EMP_ID , IXFLD=EMPLOYEE_ID , $
```

If the EMPJOB segment (EMPLOYEE-JOB record) belonged to a different subschema, the ACCESS example above would include another subschema declaration positioned between the two segment declarations.

Index Declaration Keywords for Network Record Types

The Data Adapter for IDMS/DB supports two indexing schemes: the traditional method of indexing, using the Sequential Processing Facility (SPF), and IDMS/DB Integrated Indexes. Under SPF, index entries are stored in separate index areas. As of IDMS/DB Release 10, indexes may be integrated with the database management system (DBMS).

The following keywords apply to declarations for both SPF and Integrated Indexes. For an example of an index declaration, see *CA-IDMS/DB Access File Syntax* on page 10-32.

Keyword	Description
IXSET	Is the IDMS/DB setname of the index set.
IXFLD	Is the corresponding Master File field name with FIELDTYPE=I.
IXDUP	Indicates if duplicate index values are allowed. Value can be Y or N.
IXORD	Indicates sort order of index. Value can be A (ascending) or D (descending).
IXAREA	Is the IDMS/DB area name of the index; usage varies, see below.

Note:

- This declaration is not used in LRF Access Files.
- For subschemas with SPF indexing, the IXAREA keyword is required.
- For subschemas with Integrated Indexes, the IXAREA keyword is omitted unless the index entries reside in a different area from record type being indexed.
- A non-unique descendant with an SPF index may not have descendants (immediate or several levels removed) that are related to the same index. This is due to the lack of an IDMS/DB command, which would restore currency after it has been disturbed by the record retrieval of a descendant segment. This restriction does not apply to SPF unique descendants or Integrated Index descendants.

IDMS/DB Sample File Descriptions

The following sections contain:

- The schema EMPSCHM.
- A network subschema for the schema EMPSCHM, plus corresponding Master and Access Files.
- An LRF subschema for the schema EMPSCHM, plus corresponding Master and Access Files.
- A sample of a NULL SELECT clause that creates an LRF partial record occurrence.
- A sample from a subschema that contains SPF indexes.

Note: Some samples are annotated to illustrate specific clauses.

Schema: EMPSCHM

This schema is the physical description of the IDMS/DB EMPSCHM database. It contains the following items:

- Area-to-file and area-to-DDNAME mapping.
- A CALC-based record.
- A VIA-based record.
- A sorted set ordered by the SKILL-LEVEL field.
- A sorted set ordered by Integrated Indexes using the SKILL-LEVEL field.
- An Integrated Index set ordered by the SKILL-NAME field.

```
ADD SCHEMA NAME IS EMPSCHM VERSION IS 1
SCHEMA DESCRIPTION IS 'EMPLOYEE DEMO DATABASE'
COMMENTS 'INSTALLATION: COMMONWEATHER CORPORATION'
.
ADD FILE NAME IS EMPDEMO ASSIGN TO EMPDEMO
      DEVICE TYPE IS 3380
.
      ADD FILE NAME IS INSDEMO ASSIGN TO INSDEMO
      DEVICE TYPE IS 3380
.
      ADD FILE NAME IS ORGDEMO ASSIGN TO ORGDEMO
      DEVICE TYPE IS 3380
```

.
ADD AREA NAME IS EMP-DEMO-REGION
RANGE IS 75001 THRU 75100
WITHIN FILE EMPDEMO FROM 1 THRU 100
.

ADD AREA NAME IS ORG-DEMO-REGION
RANGE IS 75151 THRU 75200
WITHIN FILE ORGDEMO FROM 1 THRU 50
.

ADD AREA NAME IS INS-DEMO-REGION
RANGE IS 75101 THRU 75150
WITHIN FILE INSDEMO FROM 1 THRU 50
.

ADD RECORD NAME IS COVERAGE
SHARE STRUCTURE OF RECORD COVERAGE VERSION IS 1
RECORD ID IS 0400
LOCATION MODE IS VIA EMP-COVERAGE SET
WITHIN AREA INS-DEMO-REGION
OFFSET 2 PAGES FOR 48 PAGES
.

ADD RECORD NAME IS DENTAL-CLAIM
SHARE STRUCTURE OF RECORD DENTAL-CLAIM VERSION IS 1
RECORD ID IS 0405
LOCATION MODE IS VIA COVERAGE-CLAIMS SET
WITHIN AREA INS-DEMO-REGION
OFFSET 2 PAGES FOR 48 PAGES
MINIMUM ROOT LENGTH IS 130 CHARACTERS
MINIMUM FRAGMENT LENGTH IS RECORD LENGTH
.

ADD RECORD NAME IS DEPARTMENT
SHARE STRUCTURE OF RECORD DEPARTMENT VERSION IS 1
RECORD ID IS 0410
LOCATION MODE IS CALC USING DEPT-ID-0410
DUPLICATES NOT ALLOWED
WITHIN AREA ORG-DEMO-REGION
OFFSET 2 PAGES FOR 48 PAGES
.

ADD RECORD NAME IS EMPLOYEE
SHARE STRUCTURE OF RECORD EMPLOYEE VERSION IS 1
RECORD ID IS 0415
LOCATION MODE IS CALC USING EMP-ID-0415
.

IDMS/DB Sample File Descriptions

```

                                DUPLICATES NOT ALLOWED
    WITHIN AREA EMP-DEMO-REGION
    OFFSET  2 PAGES FOR 98 PAGES
.

ADD RECORD NAME IS EMPOSITION
  SHARE STRUCTURE OF RECORD EMPOSITION VERSION IS 1
  RECORD ID IS 0420
  LOCATION MODE IS VIA                EMP-EMPOSITION SET
  WITHIN AREA EMP-DEMO-REGION
  OFFSET  2 PAGES FOR 98 PAGES
.

ADD RECORD NAME IS EXPERTISE
  SHARE STRUCTURE OF RECORD EXPERTISE VERSION IS 1
  RECORD ID IS 0425
  LOCATION MODE IS VIA                EMP-EXPERTISE SET
  WITHIN AREA EMP-DEMO-REGION
  OFFSET  2 PAGES FOR 98 PAGES
.

ADD RECORD NAME IS HOSPITAL-CLAIM
  SHARE STRUCTURE OF RECORD HOSPITAL-CLAIM VERSION IS 1
  RECORD ID IS 0430
  LOCATION MODE IS VIA                COVERAGE-CLAIMS SET
  WITHIN AREA INS-DEMO-REGION
  OFFSET  2 PAGES FOR 48 PAGES
.

ADD RECORD NAME IS INSURANCE-PLAN
  SHARE STRUCTURE OF RECORD INSURANCE-PLAN VERSION IS 1 |
  RECORD ID IS 0435
  LOCATION MODE IS CALC              USING INS-PLAN-CODE-0435
                                      DUPLICATES NOT ALLOWED
  WITHIN AREA INS-DEMO-REGION
  OFFSET  1 PAGE FOR 1 PAGE
.

ADD RECORD NAME IS JOB
  SHARE STRUCTURE OF RECORD JOB VERSION IS 1
  RECORD ID IS 0440
  LOCATION MODE IS CALC              USING JOB-ID-0440
                                      DUPLICATES NOT ALLOWED
  WITHIN AREA ORG-DEMO-REGION
  OFFSET  2 PAGES FOR 48 PAGES
  MINIMUM ROOT LENGTH IS CONTROL LENGTH
  MINIMUM FRAGMENT LENGTH IS RECORD LENGTH
  CALL IDMSCOMP BEFORE STORE
```


CALL IDMSCOMP BEFORE MODIFY
CALL IDMSDCOM AFTER GET

ADD RECORD NAME IS NON-HOSP-CLAIM
SHARE STRUCTURE OF RECORD NON-HOSP-CLAIM VERSION IS 1
RECORD ID IS 0445
LOCATION MODE IS VIA COVERAGE-CLAIMS SET
WITHIN AREA INS-DEMO-REGION
OFFSET 2 PAGES FOR 48 PAGES
MINIMUM ROOT LENGTH IS 248 CHARACTERS
MINIMUM FRAGMENT LENGTH IS RECORD LENGTH

ADD RECORD NAME IS OFFICE
SHARE STRUCTURE OF RECORD OFFICE VERSION IS 1
RECORD ID IS 0450
LOCATION MODE IS CALC USING OFFICE-CODE-0450
 DUPLICATES NOT ALLOWED
WITHIN AREA ORG-DEMO-REGION
OFFSET 2 PAGES FOR 48 PAGES

ADD RECORD NAME IS SKILL
SHARE STRUCTURE OF RECORD SKILL VERSION IS 1
RECORD ID IS 0455
LOCATION MODE IS CALC USING SKILL-ID-0455
 DUPLICATES NOT ALLOWED
WITHIN AREA ORG-DEMO-REGION
OFFSET 2 PAGES FOR 48 PAGES

ADD RECORD NAME IS STRUCTURE
SHARE STRUCTURE OF RECORD STRUCTURE VERSION IS 1
RECORD ID IS 0460
LOCATION MODE IS VIA MANAGES SET
WITHIN AREA EMP-DEMO-REGION
OFFSET 2 PAGES FOR 98 PAGES

ADD SET NAME IS COVERAGE-CLAIMS
ORDER IS LAST
MODE IS CHAIN LINKED TO PRIOR
OWNER IS COVERAGE
 NEXT DBKEY POSITION IS AUTO
 PRIOR DBKEY POSITION IS AUTO
MEMBER IS HOSPITAL-CLAIM
 NEXT DBKEY POSITION IS AUTO

IDMS/DB Sample File Descriptions

PRIOR DBKEY POSITION IS AUTO
MANDATORY AUTOMATIC
MEMBER IS NON-HOSP-CLAIM
NEXT DBKEY POSITION IS AUTO
PRIOR DBKEY POSITION IS AUTO
MANDATORY AUTOMATIC
MEMBER IS DENTAL-CLAIM
NEXT DBKEY POSITION IS AUTO
PRIOR DBKEY POSITION IS AUTO
MANDATORY AUTOMATIC

ADD SET NAME IS DEPT-EMPLOYEE
ORDER IS SORTED
MODE IS CHAIN LINKED TO PRIOR
OWNER IS DEPARTMENT
NEXT DBKEY POSITION IS AUTO
PRIOR DBKEY POSITION IS AUTO
MEMBER IS EMPLOYEE
NEXT DBKEY POSITION IS AUTO
PRIOR DBKEY POSITION IS AUTO
LINKED TO OWNER
OWNER DBKEY POSITION IS AUTO
OPTIONAL AUTOMATIC
ASCENDING KEY IS (EMP-LAST-NAME-0415
EMP-FIRST-NAME-0415)
DUPLICATES LAST

ADD SET NAME IS EMP-COVERAGE
ORDER IS FIRST
MODE IS CHAIN LINKED TO PRIOR
OWNER IS EMPLOYEE
NEXT DBKEY POSITION IS AUTO
PRIOR DBKEY POSITION IS AUTO
MEMBER IS COVERAGE
NEXT DBKEY POSITION IS AUTO
PRIOR DBKEY POSITION IS AUTO
LINKED TO OWNER
OWNER DBKEY POSITION IS AUTO
MANDATORY AUTOMATIC

ADD SET NAME IS EMP-EMPOSITION
ORDER IS FIRST
MODE IS CHAIN LINKED TO PRIOR
OWNER IS EMPLOYEE
NEXT DBKEY POSITION IS AUTO

```
PRIOR DBKEY POSITION IS AUTO
MEMBER IS EMPOSITION
NEXT DBKEY POSITION IS AUTO
PRIOR DBKEY POSITION IS AUTO
LINKED TO OWNER
OWNER DBKEY POSITION IS AUTO
MANDATORY AUTOMATIC
```

```
ADD SET NAME IS EMP-EXPERTISE
ORDER IS SORTED
MODE IS CHAIN LINKED TO PRIOR
OWNER IS EMPLOYEE
NEXT DBKEY POSITION IS AUTO
PRIOR DBKEY POSITION IS AUTO
MEMBER IS EXPERTISE
NEXT DBKEY POSITION IS AUTO
PRIOR DBKEY POSITION IS AUTO
LINKED TO OWNER
OWNER DBKEY POSITION IS AUTO
MANDATORY AUTOMATIC
DESCENDING KEY IS (SKILL-LEVEL-0425 )
DUPLICATES FIRST
```

```
ADD SET NAME IS EMP-NAME-NDX
ORDER IS SORTED
MODE IS INDEX BLOCK CONTAINS 40 KEYS
OWNER IS SYSTEM
WITHIN AREA EMP-DEMO-REGION
OFFSET 1 PAGE FOR 1 PAGE
MEMBER IS EMPLOYEE
INDEX DBKEY POSITION IS AUTO
OPTIONAL AUTOMATIC
ASCENDING KEY IS ( EMP-LAST-NAME-0415
EMP-FIRST-NAME-0415 )
COMPRESSED
DUPLICATES LAST
```

```
ADD SET NAME IS JOB-EMPOSITION
ORDER IS NEXT
MODE IS CHAIN LINKED TO PRIOR
OWNER IS JOB
NEXT DBKEY POSITION IS AUTO
PRIOR DBKEY POSITION IS AUTO
MEMBER IS EMPOSITION
NEXT DBKEY POSITION IS AUTO
```

IDMS/DB Sample File Descriptions

PRIOR DBKEY POSITION IS AUTO
LINKED TO OWNER
OWNER DBKEY POSITION IS AUTO
OPTIONAL MANUAL

ADD SET NAME IS JOB-TITLE-NDX
ORDER IS SORTED
MODE IS INDEX BLOCK CONTAINS 30 KEYS
OWNER IS SYSTEM
WITHIN AREA ORG-DEMO-REGION
OFFSET 1 PAGE FOR 1 PAGE
MEMBER IS JOB
INDEX DBKEY POSITION IS AUTO
OPTIONAL AUTOMATIC
ASCENDING KEY IS (TITLE-0440)
DUPLICATES NOT ALLOWED

ADD SET NAME IS MANAGES
ORDER IS NEXT
MODE IS CHAIN LINKED TO PRIOR
OWNER IS EMPLOYEE
NEXT DBKEY POSITION IS AUTO
PRIOR DBKEY POSITION IS AUTO
MEMBER IS STRUCTURE
NEXT DBKEY POSITION IS AUTO
PRIOR DBKEY POSITION IS AUTO
LINKED TO OWNER
OWNER DBKEY POSITION IS AUTO
MANDATORY AUTOMATIC

ADD SET NAME IS OFFICE-EMPLOYEE
ORDER IS SORTED
MODE IS INDEX BLOCK CONTAINS 30 KEYS
OWNER IS OFFICE
NEXT DBKEY POSITION IS AUTO
PRIOR DBKEY POSITION IS AUTO
MEMBER IS EMPLOYEE
INDEX DBKEY POSITION IS AUTO
LINKED TO OWNER
OWNER DBKEY POSITION IS AUTO
OPTIONAL AUTOMATIC
ASCENDING KEY IS (EMP-LAST-NAME-0415
EMP-FIRST-NAME-0415)
COMPRESSED
DUPLICATES LAST

```
.  
ADD SET NAME IS REPORTS-TO  
ORDER IS NEXT  
MODE IS CHAIN LINKED TO PRIOR  
OWNER IS EMPLOYE  
    NEXT DBKEY POSITION IS AUTO  
    PRIOR DBKEY POSITION IS AUTO  
MEMBER IS STRUCTURE  
    NEXT DBKEY POSITION IS AUTO  
    PRIOR DBKEY POSITION IS AUTO  
LINKED TO OWNER  
    OWNER DBKEY POSITION IS AUTO  
OPTIONAL MANUAL
```

```
.  
ADD SET NAME IS SKILL-EXPERTISE  
ORDER IS SORTED  
MODE IS INDEX BLOCK CONTAINS 30 KEYS  
OWNER IS SKILL  
    NEXT DBKEY POSITION IS AUTO  
    PRIOR DBKEY POSITION IS AUTO  
MEMBER IS EXPERTISE  
    INDEX DBKEY POSITION IS AUTO  
LINKED TO OWNER  
    OWNER DBKEY POSITION IS AUTO  
MANDATORY AUTOMATIC  
DESCENDING KEY IS ( SKILL-LEVEL-0425 )  
    DUPLICATES FIRST
```

```
.  
ADD SET NAME IS SKILL-NAME-NDX  
ORDER IS SORTED  
MODE IS INDEX BLOCK CONTAINS 30 KEYS  
OWNER IS SYSTEM  
    WITHIN AREA ORG-DEMO-REGION  
    OFFSET 1 PAGE FOR 1 PAGE  
MEMBER IS SKILL  
    INDEX DBKEY POSITION IS AUTO  
OPTIONAL AUTOMATIC  
ASCENDING KEY IS ( SKILL-NAME-0455 )  
    DUPLICATES NOT ALLOWED
```

```
.  
VALIDATE  
.
```

Network Subschema: EMPSS01

This subschema shows the network view of the schema EMPSCHM.

```
ADD SUBSCHEMA NAME IS EMPSS01
  OF SCHEMA NAME IS EMPSCHM VERSION 1

DMCL NAME IS EMPDMCL
  OF SCHEMA NAME IS EMPSCHM VERSION 1

COMMENTS 'THIS IS THE COMPLETE VIEW OF EMPSCHM'.

ADD AREA NAME IS EMP-DEMO-REGION.

ADD AREA NAME IS INS-DEMO-REGION.

ADD AREA NAME IS ORG-DEMO-REGION.

ADD RECORD NAME IS COVERAGE.

ADD RECORD NAME IS DENTAL-CLAIM.

ADD RECORD NAME IS DEPARTMENT.

ADD RECORD NAME IS EMPLOYEE.

ADD RECORD NAME IS EMPOSITION.

ADD RECORD NAME IS EXPERTISE.

ADD RECORD NAME IS HOSPITAL-CLAIM.

ADD RECORD NAME IS INSURANCE-PLAN.

ADD RECORD NAME IS JOB.

ADD RECORD NAME IS NON-HOSP-CLAIM.

ADD RECORD NAME IS OFFICE.

ADD RECORD NAME IS SKILL.

ADD RECORD NAME IS STRUCTURE.

ADD SET COVERAGE-CLAIMS.

ADD SET DEPT-EMPLOYEE.

ADD SET EMP-COVERAGE.
```

```

ADD SET EMP-EXPERTISE.

ADD SET EMP-NAME-NDX.

ADD SET EMP-EMPOSITION.

ADD SET JOB-EMPOSITION.

ADD SET JOB-TITLE-NDX.

ADD SET MANAGES.

ADD SET OFFICE-EMPLOYEE.

ADD SET REPORTS-TO.

ADD SET SKILL-EXPERTISE.

ADD SET SKILL-NAME-NDX.

GENERATE.

```

Master File for Network

This Master File corresponds to the network subschema EMPSS01.

```

FILE=EMPFULL , SUFFIX=IDMSR , $

SEGNAME=DEPT , $
  FIELDNAME=DEPT_ID , ALIAS= , USAGE=A4 , ACTUAL=A4 , $
  FIELDNAME=DEPT_NAME , ALIAS= , USAGE=A45 , ACTUAL=A45 , $
  FIELDNAME=DEPT_HEAD , ALIAS= , USAGE=A4 , ACTUAL=A4 , $
  FIELDNAME=DEPT_DBKEY , ALIAS=DBKEY , USAGE=I10 , ACTUAL=I4 , $

SEGNAME=EMPLOYEE , PARENT=DEPT , SEGTYPE=S , $
  FIELDNAME=EMP_ID , ALIAS= , USAGE=A4 , ACTUAL=A4 , $
  GROUP=EMP_NAME , ALIAS= , USAGE=A25 , ACTUAL=A25 , $
    FIELDNAME=FIRST_NAME , ALIAS= , USAGE=A10 , ACTUAL=A10 , $
    FIELDNAME=LAST_NAME , ALIAS= , USAGE=A15 , ACTUAL=A15 , $
  FIELDNAME=EMP_STREET , ALIAS= , USAGE=A20 , ACTUAL=A20 , $
  FIELDNAME=EMP_CITY , ALIAS= , USAGE=A15 , ACTUAL=A15 , $
  FIELDNAME=EMP_STATE , ALIAS= , USAGE=A2 , ACTUAL=A2 , $
  GROUP=EMP_FULL_ZIP , ALIAS= , USAGE=A9 , ACTUAL=A9 , $
    FIELDNAME=EMP_ZIP , ALIAS= , USAGE=A5 , ACTUAL=A5 , $
    FIELDNAME=EMP_ZIP_L , ALIAS= , USAGE=A4 , ACTUAL=A4 , $
  FIELDNAME=EMP_PHONE , ALIAS= , USAGE=A10 , ACTUAL=A10 , $
  FIELDNAME=STATUS , ALIAS= , USAGE=A2 , ACTUAL=A2 , $
  FIELDNAME=SOC_SEC_NUM , ALIAS= , USAGE=A9 , ACTUAL=A9 , $

```

IDMS/DB Sample File Descriptions

```
FIELDNAME=EMP_STRT_DTE , ALIAS=           , USAGE=A6YMD , ACTUAL=A6           , $
FIELDNAME=EMP_TERM_DTE , ALIAS=           , USAGE=A6YMD , ACTUAL=A6           , $
FIELDNAME=EMP_BRTH_DTE , ALIAS=           , USAGE=A6YMD , ACTUAL=A6           , $
FIELDNAME=EMP_DBKEY    , ALIAS=DBKEY      , USAGE=I10   , ACTUAL=I4           , $

SEGNAME=OFFICE , PARENT=EMPLOYE , SEGTYPE=U , $
FIELDNAME=OFF_CODE    , ALIAS=           , USAGE=A3    , ACTUAL=A3           , $
FIELDNAME=OFF_STREET  , ALIAS=           , USAGE=A20   , ACTUAL=A20          , $
FIELDNAME=OFF_CITY    , ALIAS=           , USAGE=A15   , ACTUAL=A15          , $
FIELDNAME=OFF_STATE   , ALIAS=           , USAGE=A2    , ACTUAL=A2           , $
GROUP=OFF_FULLL_ZIP   , ALIAS=           , USAGE=A9    , ACTUAL=A9           , $
    FIELDNAME=OFF_ZIP  , ALIAS=           , USAGE=A5    , ACTUAL=A5           , $
    FIELDNAME=OFF_ZIP_L , ALIAS=           , USAGE=A4    , ACTUAL=A4           , $
FIELDNAME=O_PHONES    , ALIAS=           , USAGE=A21   , ACTUAL=A21          , $
FIELDNAME=OFF_AREA_CDE , ALIAS=           , USAGE=A3    , ACTUAL=A3           , $
FIELDNAME=SPEED_DIAL  , ALIAS=           , USAGE=A3    , ACTUAL=A3           , $

SEGNAME=PHONES , PARENT=OFFICE , SEGTYPE=S , OCCURS=3 , POSITION=O_PHONES , $
FIELDNAME=OFF_PHONE   , ALIAS=           , USAGE=A7    , ACTUAL=A7           , $
FIELDNAME=LINE_NO     , ALIAS=ORDER      , USAGE=I4    , ACTUAL=I4           , $

SEGNAME=STRUCTUR , PARENT=EMPLOYE , SEGTYPE=S , $
FIELDNAME=STRUCTURE_CD , ALIAS=           , USAGE=A2    , ACTUAL=A2           , $
FIELDNAME=STRUCTURE_DT , ALIAS=           , USAGE=A6YMD , ACTUAL=A6           , $

SEGNAME=SUBORDS , PARENT=STRUCTUR , SEGTYPE=U , $
FIELDNAME=SUB_ID      , ALIAS=           , USAGE=A4    , ACTUAL=A4           , $
GROUP=SUB_NAME        , ALIAS=           , USAGE=A25   , ACTUAL=A25          , $
    FIELDNAME=SUB_F_NAME , ALIAS=           , USAGE=A10   , ACTUAL=A10          , $
    FIELDNAME=SUB_L_NAME , ALIAS=           , USAGE=A15   , ACTUAL=A15          , $
FIELDNAME=SUB_STREET  , ALIAS=           , USAGE=A20   , ACTUAL=A20          , $
FIELDNAME=SUB_CITY    , ALIAS=           , USAGE=A15   , ACTUAL=A15          , $
FIELDNAME=SUB_STATE   , ALIAS=           , USAGE=A2    , ACTUAL=A2           , $
GROUP=SUB_FULLL_ZIP   , ALIAS=           , USAGE=A9    , ACTUAL=A9           , $
    FIELDNAME=SUB_ZIP    , ALIAS=           , USAGE=A5    , ACTUAL=A5           , $
    FIELDNAME=SUB_ZIP_L  , ALIAS=           , USAGE=A4    , ACTUAL=A4           , $
FIELDNAME=SUB_PHONE   , ALIAS=           , USAGE=A10   , ACTUAL=A10          , $
FIELDNAME=SUB_STATUS  , ALIAS=           , USAGE=A2    , ACTUAL=A2           , $
FIELDNAME=SUB_SSN     , ALIAS=           , USAGE=A9    , ACTUAL=A9           , $
FIELDNAME=SUB_STRT_DTE , ALIAS=           , USAGE=A6YMD , ACTUAL=A6           , $
FIELDNAME=SUB_TERM_DTE , ALIAS=           , USAGE=A6YMD , ACTUAL=A6           , $
FIELDNAME=SUB_BRTH_DTE , ALIAS=           , USAGE=A6YMD , ACTUAL=A6           , $

SEGNAME=EMPOSIT , PARENT=EMPLOYE , SEGTYPE=S , $
FIELDNAME=POS_STRT_DTE , ALIAS=           , USAGE=A6YMD , ACTUAL=A6           , $
FIELDNAME=POS_FIN_DTE  , ALIAS=           , USAGE=A6YMD , ACTUAL=A6           , $
FIELDNAME=SALARY_GRADE , ALIAS=           , USAGE=P4    , ACTUAL=Z2           , $
FIELDNAME=SALARY_AMT   , ALIAS=           , USAGE=P10.2 , ACTUAL=P5           , $
```



```

FIELDNAME=BONUS_PCT ,ALIAS= ,USAGE=P4 ,ACTUAL=P2 ,$,
FIELDNAME=COMMISS_PCT ,ALIAS= ,USAGE=P4 ,ACTUAL=P2 ,$,
FIELDNAME=OVERTIME_PCT,ALIAS= ,USAGE=P5.2 ,ACTUAL=P2 ,$,

SEGNAME=JOB, PARENT=EMPOSIT, SEGTYPE=U, $
FIELDNAME=JOB_ID ,ALIAS= ,USAGE=A4 ,ACTUAL=A4 ,$,
FIELDNAME=TITLE ,ALIAS= ,USAGE=A20 ,ACTUAL=A20 ,$,
FIELDTYPE=I, $
DEFINE SHORTTITLE/A10 = EDIT(JTIT, '9999999999$'); ,$,
FIELDNAME=JOB_DESC ,ALIAS= ,USAGE=A120 ,ACTUAL=A120 ,$,
FIELDNAME=REQUIREMENTS,ALIAS= ,USAGE=A120 ,ACTUAL=A120 ,$,
FIELDNAME=MIN_SALARY ,ALIAS= ,USAGE=P12.2 ,ACTUAL=Z8 ,$,
FIELDNAME=MAX_SALARY ,ALIAS= ,USAGE=P12.2 ,ACTUAL=Z8 ,$,
FIELDNAME=SAL_GRADE_1 ,ALIAS= ,USAGE=P4 ,ACTUAL=Z2 ,$,
FIELDNAME=SAL_GRADE_2 ,ALIAS= ,USAGE=P4 ,ACTUAL=Z2 ,$,
FIELDNAME=SAL_GRADE_3 ,ALIAS= ,USAGE=P4 ,ACTUAL=Z2 ,$,
FIELDNAME=SAL_GRADE_4 ,ALIAS= ,USAGE=P4 ,ACTUAL=Z2 ,$,
FIELDNAME=POSITION_NUM,ALIAS= ,USAGE=P4 ,ACTUAL=Z3 ,$,
FIELDNAME=NUM_OPEN ,ALIAS= ,USAGE=P4 ,ACTUAL=Z3 ,$,

SEGNAME=EXPERTSE, PARENT=EMPLOYE, SEGTYPE=S, $
FIELDNAME=SKILL_LEVEL ,ALIAS= ,USAGE=A2 ,ACTUAL=A2 ,$,
FIELDNAME=EXPERT_DTE ,ALIAS= ,USAGE=A6YMD ,ACTUAL=A6 ,$,

SEGNAME=SKILL, PARENT=EXPERTSE, SEGTYPE=U, $
FIELDNAME=SKILL_ID ,ALIAS= ,USAGE=A4 ,ACTUAL=A4 ,$,
FIELDNAME=SKILL_NAME ,ALIAS= ,USAGE=A12 ,ACTUAL=A12 ,$,
FIELDTYPE=I, $
FIELDNAME=SKILL_DESC ,ALIAS= ,USAGE=A60 ,ACTUAL=A60 ,$,

SEGNAME=COVERAGE, PARENT=EMPLOYE, SEGTYPE=S, $
FIELDNAME=COV_SEL_DT ,ALIAS= ,USAGE=I6YMD ,ACTUAL=Z6 ,$,
FIELDNAME=COV_TERM_DTE,ALIAS= ,USAGE=A6YMD ,ACTUAL=A6 ,$,
FIELDNAME=COVER_TYPE ,ALIAS= ,USAGE=A1 ,ACTUAL=A1 ,$,
FIELDNAME=COV_CODE ,ALIAS= ,USAGE=A3 ,ACTUAL=A3 ,$,

SEGNAME=HOSPITAL, PARENT=COVERAGE, SEGTYPE=S, $
FIELDNAME=H_CLAIM_DTE ,ALIAS= ,USAGE=I6YMD ,ACTUAL=Z6 ,$,
FIELDNAME=H_FIRST_NAME,ALIAS= ,USAGE=A10 ,ACTUAL=A10 ,$,
FIELDNAME=H_LAST_NAME ,ALIAS= ,USAGE=A15 ,ACTUAL=A15 ,$,
FIELDNAME=H_BIRTH_DTE ,ALIAS= ,USAGE=I6YMD ,ACTUAL=Z6 ,$,
FIELDNAME=H_SEX ,ALIAS= ,USAGE=A1 ,ACTUAL=A1 ,$,
FIELDNAME=H_RELATED_BY,ALIAS= ,USAGE=A10 ,ACTUAL=A10 ,$,
FIELDNAME=HOSP_NAME ,ALIAS= ,USAGE=A25 ,ACTUAL=A25 ,$,
FIELDNAME=HOSP_STREET ,ALIAS= ,USAGE=A20 ,ACTUAL=A20 ,$,
FIELDNAME=HOSP_CITY ,ALIAS= ,USAGE=A15 ,ACTUAL=A15 ,$,
FIELDNAME=HOSP_STATE ,ALIAS= ,USAGE=A2 ,ACTUAL=A2 ,$,
GROUP=HOSP_FUL_ZIP ,ALIAS= ,USAGE=A9 ,ACTUAL=A9 ,$,

```

IDMS/DB Sample File Descriptions

```

        FIELDNAME=HOSP_ZIP ,ALIAS= ,USAGE=A5 ,ACTUAL=A5 ,$,
        FIELDNAME=HOSP_ZIP_L,ALIAS= ,USAGE=A4 ,ACTUAL=A4 ,$,
        FIELDNAME=ADMITTED ,ALIAS= ,USAGE=I6YMD,ACTUAL=Z6 ,$,
        FIELDNAME=DISCHARGED ,ALIAS= ,USAGE=I6YMD,ACTUAL=Z6 ,$,
        FIELDNAME=H_DIAGNOSIS1,ALIAS= ,USAGE=A60 ,ACTUAL=A60 ,$,
        FIELDNAME=H_DIAGNOSIS2,ALIAS= ,USAGE=A60 ,ACTUAL=A60 ,$,
        FIELDNAME=WARD_DAYS ,ALIAS= ,USAGE=P5 ,ACTUAL=P3 ,$,
        FIELDNAME=WARD_RATE ,ALIAS= ,USAGE=P10.2,ACTUAL=P5 ,$,
        FIELDNAME=WARD_TOTAL ,ALIAS= ,USAGE=P10.2,ACTUAL=P5 ,$,
        FIELDNAME=SEMI_DAYS ,ALIAS= ,USAGE=P5 ,ACTUAL=P3 ,$,
        FIELDNAME=SEMI_RATE ,ALIAS= ,USAGE=P10.2,ACTUAL=P5 ,$,
        FIELDNAME=SEMI_TOTAL ,ALIAS= ,USAGE=P10.2,ACTUAL=P5 ,$,
        FIELDNAME=DELIVERY_TOT,ALIAS= ,USAGE=P10.2,ACTUAL=P5 ,$,
        FIELDNAME=ANESTHES_TOT,ALIAS= ,USAGE=P10.2,ACTUAL=P5 ,$,
        FIELDNAME=LAB_TOT ,ALIAS= ,USAGE=P10.2,ACTUAL=P5 ,$,
        FIELDNAME= ,ALIAS= ,USAGE=A4 ,ACTUAL=A4 ,$,
        FIELDNAME=CLAIM_MONTH ,ALIAS=CMO ,USAGE=I2 ,ACTUAL=Z2 ,$,

SEGNAME=NON_HOSP ,SEGTYPE=S ,PARENT=COVERAGE , $
        FIELDNAME=N_CLAIM_DTE ,ALIAS= ,USAGE=I6YMD,ACTUAL=Z6 ,$,
        FIELDNAME=N_FIRST_NAME,ALIAS= ,USAGE=A10 ,ACTUAL=A10 ,$,
        FIELDNAME=N_LAST_NAME ,ALIAS= ,USAGE=A15 ,ACTUAL=A15 ,$,
        FIELDNAME=N_BIRTH_DTE ,ALIAS= ,USAGE=I6YMD,ACTUAL=Z6 ,$,
        FIELDNAME=N_SEX ,ALIAS= ,USAGE=A1 ,ACTUAL=A1 ,$,
        FIELDNAME=N_RELATED_BY,ALIAS= ,USAGE=A10 ,ACTUAL=A10 ,$,
        FIELDNAME=PHYS_FNAME ,ALIAS= ,USAGE=A10 ,ACTUAL=A10 ,$,
        FIELDNAME=PHYS_LNAME ,ALIAS= ,USAGE=A15 ,ACTUAL=A15 ,$,
        FIELDNAME=PHYS_STREET ,ALIAS= ,USAGE=A20 ,ACTUAL=A20 ,$,
        FIELDNAME=PHYS_CITY ,ALIAS= ,USAGE=A15 ,ACTUAL=A15 ,$,
        FIELDNAME=PHYS_STATE ,ALIAS= ,USAGE=A2 ,ACTUAL=A2 ,$,
        GROUP=PHYS_FUL_ZIP ,ALIAS= ,USAGE=A9 ,ACTUAL=A9 ,$,
        FIELDNAME=PHYS_ZIP ,ALIAS= ,USAGE=A5 ,ACTUAL=A5 ,$,
        FIELDNAME=PHYS_ZIP_L,ALIAS= ,USAGE=A4 ,ACTUAL=A4 ,$,
        FIELDNAME=PHYS_ID ,ALIAS= ,USAGE=P6 ,ACTUAL=Z6 ,$,
        FIELDNAME=P_DIAGNOSIS1,ALIAS= ,USAGE=A60 ,ACTUAL=A60 ,$,
        FIELDNAME=P_DIAGNOSIS2,ALIAS= ,USAGE=A60 ,ACTUAL=A60 ,$,
        FIELDNAME=P_NO_OF_PROC,ALIAS= ,USAGE=I2 ,ACTUAL=I2 ,$,
        FIELDNAME= ,ALIAS= ,USAGE=A1 ,ACTUAL=A1 ,$,

SEGNAME=PHYSCHRG ,SEGTYPE=S ,PARENT=NON_HOSP ,OCCURS=P_NO_OF_PROC , $
        FIELDNAME=P_SERVICE_DT,ALIAS= ,USAGE=I6YMD,ACTUAL=Z6 ,$,
        FIELDNAME=PHYS_PROC_CD,ALIAS= ,USAGE=P4 ,ACTUAL=Z4 ,$,
        FIELDNAME=P_SERV_DESC ,ALIAS= ,USAGE=A60 ,ACTUAL=A60 ,$,
        FIELDNAME=PHYS_FEE ,ALIAS= ,USAGE=P11.2,ACTUAL=P5 ,$,
        FIELDNAME= ,ALIAS= ,USAGE=A1 ,ACTUAL=A1 ,$,
        FIELDNAME=PHYS_CHRG_NO,ALIAS=ORDER ,USAGE=I4 ,ACTUAL=I4 ,$,

SEGNAME=DENTAL ,SEGTYPE=S ,PARENT=COVERAGE , $

```

```

FIELDNAME=D_CLAIM_DTE ,ALIAS= ,USAGE=I6YMD,ACTUAL=Z6 ,$,
FIELDNAME=D_FIRST_NAME,ALIAS= ,USAGE=A10 ,ACTUAL=A10 ,$,
FIELDNAME=D_LAST_NAME ,ALIAS= ,USAGE=A15 ,ACTUAL=A15 ,$,
FIELDNAME=D_BIRTH_DTE ,ALIAS= ,USAGE=I6YMD,ACTUAL=Z6 ,$,
FIELDNAME=D_SEX ,ALIAS= ,USAGE=A1 ,ACTUAL=A1 ,$,
FIELDNAME=D_RELATED_BY,ALIAS= ,USAGE=A10 ,ACTUAL=A10 ,$,
FIELDNAME=DENT_FNAME ,ALIAS= ,USAGE=A10 ,ACTUAL=A10 ,$,
FIELDNAME=DENT_LNAME ,ALIAS= ,USAGE=A15 ,ACTUAL=A15 ,$,
FIELDNAME=DENT_STREET ,ALIAS= ,USAGE=A20 ,ACTUAL=A20 ,$,
FIELDNAME=DENT_CITY ,ALIAS= ,USAGE=A15 ,ACTUAL=A15 ,$,
FIELDNAME=DENT_STATE ,ALIAS= ,USAGE=A2 ,ACTUAL=A2 ,$,
GROUP=DENT_FUL_ZIP ,ALIAS= ,USAGE=A9 ,ACTUAL=A9 ,$,
    FIELDNAME=DENT_ZIP ,ALIAS= ,USAGE=A5 ,ACTUAL=A5 ,$,
    FIELDNAME=DENT_ZIP_L,ALIAS= ,USAGE=A4 ,ACTUAL=A4 ,$,
FIELDNAME=DENT_LICENSE,ALIAS= ,USAGE=P6 ,ACTUAL=Z6 ,$,
FIELDNAME=D_NO_OF_PROC,ALIAS= ,USAGE=I2 ,ACTUAL=I2 ,$,
FIELDNAME= ,ALIAS= ,USAGE=A3 ,ACTUAL=A3 ,$,

SEGNAME=DENTCHRG ,SEGTYPE=S ,PARENT=DENTAL ,OCCURS=D_NO_OF_PROC , $
    FIELDNAME=TOOTH_NUM ,ALIAS= ,USAGE=P2 ,ACTUAL=Z2 ,$,
    FIELDNAME=D_SERVICE_DT,ALIAS= ,USAGE=A6YMD,ACTUAL=A6 ,$,
    FIELDNAME=DENT_PROC_CD,ALIAS= ,USAGE=P4 ,ACTUAL=Z4 ,$,
    FIELDNAME=D_SERV_DESC ,ALIAS= ,USAGE=A60 ,ACTUAL=A60 ,$,
    FIELDNAME=DENT_FEE ,ALIAS= ,USAGE=P11.2,ACTUAL=P5 ,$,
    FIELDNAME= ,ALIAS= ,USAGE=A3 ,ACTUAL=A3 ,$,
    FIELDNAME=DENT_CHRG_NO,ALIAS=ORDER ,USAGE=I9 ,ACTUAL=I4 ,$,

SEGNAME=INSURNCE ,PARENT=COVERAGE ,SEGTYPE=U , $
    FIELDNAME=INS_PLAN_CDE,ALIAS= ,USAGE=A3 ,ACTUAL=A3 ,$,
    FIELDNAME=INS_CO_NAME ,ALIAS= ,USAGE=A45 ,ACTUAL=A45 ,$,
    FIELDNAME=INS_STREET ,ALIAS= ,USAGE=A20 ,ACTUAL=A20 ,$,
    FIELDNAME=INS_CITY ,ALIAS= ,USAGE=A15 ,ACTUAL=A15 ,$,
    FIELDNAME=INS_STATE ,ALIAS= ,USAGE=A2 ,ACTUAL=A2 ,$,
    GROUP=INS_FULL_ZIP ,ALIAS= ,USAGE=A9 ,ACTUAL=A9 ,$,
        FIELDNAME=INS_ZIP ,ALIAS= ,USAGE=A5 ,ACTUAL=A5 ,$,
        FIELDNAME=INS_ZIP_L ,ALIAS= ,USAGE=A4 ,ACTUAL=A4 ,$,
    FIELDNAME=INS_PHONE ,ALIAS= ,USAGE=A10 ,ACTUAL=A10 ,$,
    FIELDNAME=INS_GROUP_NO,ALIAS= ,USAGE=A6 ,ACTUAL=A6 ,$,
    FIELDNAME=DEDUCT ,ALIAS= ,USAGE=P12.2,ACTUAL=P5 ,$,
    FIELDNAME=MAX_LIFE_CST,ALIAS= ,USAGE=P12.2,ACTUAL=P5 ,$,
    FIELDNAME=FAMILY_COST ,ALIAS= ,USAGE=P12.2,ACTUAL=P5 ,$,
    FIELDNAME=DEPENDNT_CST,ALIAS= ,USAGE=P12.2,ACTUAL=P5 ,$,

```

Access File for Network

This Access File is associated with the network subschema EMPSS01 and corresponds to its Master File.

```
SSCHEMA=EMPSS01,RELEASE=15,MODE=DML,TRACE=NO,READY=,$

SEGNAM=DEPT,RECORD=DEPARTMENT,AREA=ORG-DEMO-REGION,
  CLCFLD=DEPT_ID,CLCDUP=N,$

SEGNAM=EMPLOYEE,RECORD=EMPLOYEE,AREA=EMP-DEMO-REGION,
  CLCFLD=EMP_ID,CLCDUP=N,ACCESS=SET,SETNAME=DEPT-EMPLOYEE,
  SETMBR=OA,GETOWN=Y,MULTMBR=N,$

SEGNAM=OFFICE,RECORD=OFFICE,AREA=ORG-DEMO-REGION,
  CLCFLD=OFF_CODE,CLCDUP=N,ACCESS=SET,SETNAME=OFFICE-EMPLOYEE,
  SETMBR=OA,GETOWN=Y,MULTMBR=N,$

SEGNAM=STRUCTUR,RECORD=STRUCTURE,AREA=EMP-DEMO-REGION,
  ACCESS=SET,SETNAME=MANAGES,SETMBR=MA,GETOWN=Y,MULTMBR=N,$

SEGNAM=SUBORDS,RECORD=EMPLOYEE,AREA=EMP-DEMO-REGION,
  CLCFLD=SUB_ID,CLCDUP=N,ACCESS=SET,SETNAME=REPORTS-TO,
  SETMBR=OM,GETOWN=Y,MULTMBR=N,$

SEGNAM=EMPOSIT,RECORD=EMPOSITION,AREA=EMP-DEMO-REGION,
  ACCESS=SET,SETNAME=EMP-EMPOSITION,SETMBR=MA,GETOWN=Y,MULTMBR=N,$

SEGNAM=JOB,RECORD=JOB,AREA=ORG-DEMO-REGION,
  CLCFLD=JOB_ID,CLCDUP=N,ACCESS=SET,SETNAME=JOB-EMPOSITION,
  SETMBR=OM,GETOWN=Y,MULTMBR=N,SEQFIELD=TITLE,$

SEGNAM=EXPERTSE,RECORD=EXPERTISE,AREA=EMP-DEMO-REGION,
  ACCESS=SET,SETNAME=EMP-EXPERTISE,KEYFLD=SKILL_LEVEL,SETORD=D,
  SETDUP=Y,SETMBR=MA,GETOWN=Y,MULTMBR=N,$

SEGNAM=SKILL,RECORD=SKILL,AREA=ORG-DEMO-REGION,
  CLCFLD=SKILL_ID,CLCDUP=N,ACCESS=SET,SETNAME=SKILL-EXPERTISE,
  KEYFLD=SKILL_LEVEL,SETORD=D,SETDUP=Y,
  SETMBR=MA,GETOWN=Y,MULTMBR=N,SEQFIELD=SKILL_NAME,$

SEGNAM=COVERAGE,RECORD=COVERAGE,AREA=INS-DEMO-REGION,
  ACCESS=SET,SETNAME=EMP-COVERAGE,SETMBR=MA,GETOWN=Y,MULTMBR=N,$

SEGNAM=HOSPITAL,RECORD=HOSPITAL-CLAIM,AREA=INS-DEMO-REGION,
  ACCESS=SET,SETNAME=COVERAGE-CLAIMS,SETMBR=MA,GETOWN=Y,MULTMBR=Y,$

SEGNAM=NON_HOSP,RECORD=NON-HOSP-CLAIM,AREA=INS-DEMO-REGION,
  ACCESS=SET,SETNAME=COVERAGE-CLAIMS,SETMBR=MA,GETOWN=Y,MULTMBR=Y,$
```

```
SEGNAM=DENTAL, RECORD=DENTAL-CLAIM, AREA=INS-DEMO-REGION,
ACCESS=SET, SETNAME=COVERAGE-CLAIMS, SETMBR=MA, GETOWN=Y, MULTMBR=Y, $
```

```
IXSET=JOB-TITLE-NDX, IXFLD=TITLE, IXDUP=N, IXORD=A,
IXAREA=INS-DEMO-REGION, $
```

```
IXSET=SKILL-NAME-NDX, IXFLD=SKILL_NAME, IXDUP=N, IXORD=D,
IXAREA=EMP-DEMO-REGION, $
```

```
SEGNAM=INSURNCE, RECORD=INSURANCE-PLAN, AREA=INS-DEMO-REGION,
CLCFLD=INS_PLAN_CDE, CLCDUP=N, ACCESS=CLC, KEYFLD=COV_CODE, $
```

LRF Subschema: EMPSS02

This subschema shows the LRF view of the schema EMPSCHEM. It contains the following items:

- Physical record types that are used to create logical records.
- A SELECT clause for CALC access.
- A SELECT ELEMENT clause.
- A SELECT NULL clause.
- A SELECT INDEX clause.

```
ADD SUBSCHEMA NAME IS EMPSS02
  OF SCHEMA NAME IS EMPSCHEM VERSION 1
  USAGE IS LR
```

```
DMCL NAME IS EMPDMCL
  OF SCHEMA NAME IS EMPSCHEM VERSION 1
```

```
COMMENTS 'THIS IS THE COMPLETE VIEW OF EMPSCHEM'.
```

```
ADD AREA NAME IS EMP-DEMO-REGION.
```

```
ADD AREA NAME IS ORG-DEMO-REGION.
```

```
ADD RECORD NAME IS DEPARTMENT.
```

```
ADD RECORD NAME IS EMPLOYEE.
```

```
ADD RECORD NAME IS EMPOSITION.
```

```
ADD RECORD NAME IS JOB.
```

```
ADD SET DEPT-EMPLOYEE.
```

IDMS/DB Sample File Descriptions

ADD SET EMP-NAME-NDX.

ADD SET EMP-EMPOSITION.

ADD SET JOB-EMPOSITION.

ADD SET JOB-TITLE-NDX.

ADD

LOGICAL RECORD NAME IS DEPT-EMP-POS
ELEMENTS ARE DEPARTMENT
EMPLOYEE
EMPOSITION.

ADD

PATH-GROUP NAME IS OBTAIN DEPT-EMP-POS
SELECT FOR FIELDNAME-EQ DEPT-ID-0410
OBTAIN DEPARTMENT
WHERE CALCKEY EQ DEPT-ID-0410 OF REQUEST
IF DEPT-EMPLOYEE IS NOT EMPTY
OBTAIN EACH EMPLOYEE WITHIN DEPT-EMPLOYEE
IF EMP-EMPOSITION IS NOT EMPTY
OBTAIN EACH EMPOSITION WITHIN EMP-EMPOSITION
SELECT FOR FIELDNAME-EQ EMP-ID-0415
OBTAIN EMPLOYEE
WHERE CALCKEY EQ EMP-ID-0415 OF REQUEST
IF DEPT-EMPLOYEE MEMBER
OBTAIN OWNER WITHIN DEPT-EMPLOYEE
IF EMP-EMPOSITION IS NOT EMPTY
OBTAIN EACH EMPOSITION WITHIN EMP-EMPOSITION
SELECT FOR ELEMENT DEPARTMENT
OBTAIN EACH DEPARTMENT WITHIN ORG-DEMO-REGION
IF DEPT-EMPLOYEE IS NOT EMPTY
OBTAIN EACH EMPLOYEE WITHIN DEPT-EMPLOYEE
IF EMP-EMPOSITION IS NOT EMPTY
OBTAIN EACH EMPOSITION WITHIN EMP-EMPOSITION
SELECT FOR ELEMENT EMPLOYEE
OBTAIN EACH EMPLOYEE WITHIN EMP-DEMO-REGION
IF DEPT-EMPLOYEE MEMBER
OBTAIN OWNER WITHIN DEPT-EMPLOYEE
IF EMP-EMPOSITION IS NOT EMPTY
OBTAIN EACH EMPOSITION WITHIN EMP-EMPOSITION
SELECT FOR ELEMENT EMPOSITION
OBTAIN EACH EMPOSITION WITHIN EMP-DEMO-REGION
OBTAIN OWNER WITHIN EMP-EMPOSITION
IF DEPT-EMPLOYEE MEMBER

```

      OBTAIN OWNER WITHIN DEPT-EMPLOYEE
SELECT
      OBTAIN EACH DEPARTMENT WITHIN ORG-DEMO-REGION
      IF DEPT-EMPLOYEE IS NOT EMPTY
      OBTAIN EACH EMPLOYEE WITHIN DEPT-EMPLOYEE
      IF EMP-EMPOSITION IS NOT EMPTY
      OBTAIN EACH EMPOSITION WITHIN EMP-EMPOSITION.

ADD
      LOGICAL RECORD NAME IS JOB-EMPOSITION
      ELEMENTS ARE JOB
          EMPOSITION.

ADD

PATH-GROUP NAME IS OBTAIN JOB-EMPOSITION
      SELECT FOR FIELDNAME-EQ JOB-ID-0440
          OBTAIN JOB
              WHERE CALCKEY EQ JOB-ID-0440 OF REQUEST
          IF JOB-EMPOSITION IS NOT EMPTY
          OBTAIN EACH EMPOSITION WITHIN JOB-EMPOSITION

      SELECT USING INDEX JOB-TITLE-NDX
          FOR FIELDNAME TITLE-0440
          OBTAIN EACH JOB USING INDEX
          IF JOB-EMPOSITION IS NOT EMPTY
          OBTAIN EACH EMPOSITION WITHIN JOB-EMPOSITION
      SELECT FOR FIELDNAME START-DATE-0420
          OBTAIN EACH EMPOSITION WITHIN EMP-DEMO-REGION
          IF JOB-EMPOSITION MEMBER
          OBTAIN OWNER WITHIN JOB-EMPOSITION
      SELECT FOR ELEMENT JOB
          OBTAIN EACH JOB WITHIN ORG-DEMO-REGION
          IF JOB-EMPOSITION IS NOT EMPTY
          OBTAIN EACH EMPOSITION WITHIN JOB-EMPOSITION
      SELECT FOR ELEMENT EMPOSITION
          OBTAIN EACH EMPOSITION WITHIN EMP-DEMO-REGION
          IF JOB-EMPOSITION MEMBER
          OBTAIN OWNER WITHIN JOB-EMPOSITION
      SELECT
          OBTAIN EACH JOB WITHIN ORG-DEMO-REGION
              ON 0307 CLEAR RETURN LR-NOT-FOUND
              ON 0000 NEXT
          IF JOB-EMPOSITION IS NOT EMPTY
              ON 0000 ITERATE
              ON 1601 NEXT
          OBTAIN EACH EMPOSITION WITHIN JOB-EMPOSITION
              ON 0000 NEXT

```

ON 0307 ITERATE.

GENERATE.

Master File for LRF

FILE=EMPDATA , SUFFIX=IDMSR , \$

SEGNAME=DEPTEMPO , \$

FIELD=DEPT_ID	, ALIAS=	, USAGE=A4	, ACTUAL=A4	, \$
FIELD=DEPT_NAME	, ALIAS=	, USAGE=A45	, ACTUAL=A45	, \$
FIELD=DEPT_HEAD	, ALIAS=	, USAGE=A4	, ACTUAL=A4	, \$
FIELD=	, ALIAS=FILL.END	, USAGE=A3	, ACTUAL=A3	, \$
FIELD=EMP_ID	, ALIAS=	, USAGE=A4	, ACTUAL=A4	, \$
GROUP=EMP_NAME	, ALIAS=	, USAGE=A25	, ACTUAL=A25	, \$
FIELD=FIRST_NAME	, ALIAS=	, USAGE=A10	, ACTUAL=A10	, \$
FIELD=LAST_NAME	, ALIAS=	, USAGE=A15	, ACTUAL=A15	, \$
FIELD=EMP_STREET	, ALIAS=	, USAGE=A20	, ACTUAL=A20	, \$
FIELD=EMP_CITY	, ALIAS=	, USAGE=A15	, ACTUAL=A15	, \$
FIELD=EMP_STATE	, ALIAS=	, USAGE=A2	, ACTUAL=A2	, \$
GROUP=EMP_FULL_ZIP	, ALIAS=	, USAGE=A9	, ACTUAL=A9	, \$
FIELD=EMP_ZIP	, ALIAS=	, USAGE=A5	, ACTUAL=A5	, \$
FIELD=EMP_ZIP_L	, ALIAS=	, USAGE=A4	, ACTUAL=A4	, \$
FIELD=EMP_PHONE	, ALIAS=	, USAGE=A10	, ACTUAL=A10	, \$
FIELD=STATUS	, ALIAS=	, USAGE=A2	, ACTUAL=A2	, \$
FIELD=SOC_SEC_NUM	, ALIAS=	, USAGE=A9	, ACTUAL=A9	, \$
FIELD=EMP_STRT_DTE	, ALIAS=	, USAGE=A6YMD	, ACTUAL=A6	, \$
FIELD=EMP_TERM_DTE	, ALIAS=	, USAGE=A6YMD	, ACTUAL=A6	, \$
FIELD=EMP_BRTH_DTE	, ALIAS=	, USAGE=A6YMD	, ACTUAL=A6	, \$
FIELD=	, ALIAS=FILL.END	, USAGE=A6	, ACTUAL=A6	, \$
FIELD=POS_STRT_DT1	, ALIAS=	, USAGE=A6YMD	, ACTUAL=A6	, \$
FIELD=POS_FIN_DT1	, ALIAS=	, USAGE=A6YMD	, ACTUAL=A6	, \$
FIELD=SALARY_GRAD1	, ALIAS=	, USAGE=P4	, ACTUAL=Z2	, \$
FIELD=SALARY_AMT1	, ALIAS=	, USAGE=P10.2	, ACTUAL=P5	, \$
FIELD=BONUS_PCT1	, ALIAS=	, USAGE=P4	, ACTUAL=P2	, \$
FIELD=COMMIS_PCT1	, ALIAS=	, USAGE=P4	, ACTUAL=P2	, \$
FIELD=OVERTIM_PCT1	, ALIAS=	, USAGE=P5.2	, ACTUAL=P2	, \$

SEGNAME=JOBPOS , PARENT=DEPTEMPO , SEGTYPE=U , \$

FIELD=JOB_ID	, ALIAS=	, USAGE=A4	, ACTUAL=A4	, \$
FIELD=TITLE	, ALIAS=	, USAGE=A20	, ACTUAL=A20	, \$
FIELD=JOB_DESC	, ALIAS=	, USAGE=A120	, ACTUAL=A120	, \$
FIELD=REQUIREMENTS	, ALIAS=	, USAGE=A120	, ACTUAL=A120	, \$
FIELD=MIN_SALARY	, ALIAS=	, USAGE=P12.2	, ACTUAL=Z8	, \$
FIELD=MAX_SALARY	, ALIAS=	, USAGE=P12.2	, ACTUAL=Z8	, \$
FIELD=SAL_GRADE_1	, ALIAS=	, USAGE=P4	, ACTUAL=Z2	, \$
FIELD=SAL_GRADE_2	, ALIAS=	, USAGE=P4	, ACTUAL=Z2	, \$
FIELD=SAL_GRADE_3	, ALIAS=	, USAGE=P4	, ACTUAL=Z2	, \$
FIELD=SAL_GRADE_4	, ALIAS=	, USAGE=P4	, ACTUAL=Z2	, \$


```

FIELD=POSITION_NUM, ALIAS=          , USAGE=P4      , ACTUAL=Z3    , $
FIELD=NUM_OPEN      , ALIAS=          , USAGE=P4      , ACTUAL=Z3    , $
FIELD=              , ALIAS=FILL.END, USAGE=A2      , ACTUAL=A2    , $
FIELD=POS_STRT_DT2 , ALIAS=          , USAGE=A6YMD  , ACTUAL=A6    , $
FIELD=POS_FIN_DT2  , ALIAS=          , USAGE=A6YMD  , ACTUAL=A6    , $
FIELD=SALARY_GRAD2 , ALIAS=          , USAGE=P4      , ACTUAL=Z2    , $
FIELD=SALARY_AMT2  , ALIAS=          , USAGE=P10.2  , ACTUAL=P5    , $
FIELD=BONUS_PCT2   , ALIAS=          , USAGE=P4      , ACTUAL=P2    , $
FIELD=COMMIS_PCT2  , ALIAS=          , USAGE=P4      , ACTUAL=P2    , $
FIELD=OVERTIM_PCT2, ALIAS=          , USAGE=P5.2   , ACTUAL=P2    , $

```

Access File for LRF

This Access File is associated with LRF subschema EMPSS02, and corresponds to its Master File.

```

SSCHEMA=EMPSS02 , RELEASE=15 , MODE=LR , TRACE=PARMS , READY=ALL , $
SEGNAM=DEPTEMPO , RECORD=DEPT-EMP-POS , AREA=EMP-DEMO-REGION , LR=Y , $
SEGNAM=JOBPOS , RECORD=JOB-EMPOSITION , AREA=ORG-DEMO-REGION , LR=Y ,
ACCESS=LR , KEYFLD=POS_STRT_DT1 , IXFLD=POS_STRT_DT2 , $

```

Sample of a Partial LRF Record

The following is an example of a NULL SELECT clause that creates a partial record by returning a user-defined record code. The adapter does not support this user-defined code or any status code other than LR-FOUND or LR-NOT-FOUND.

```

SELECT
  OBTAIN EACH JOB WITHIN ORG-DEMO-REGION
  IF JOB-EMPOSITION IS NOT EMPTY
  ON 0000 RETURN NO-POS-FOR-JOB
  OBTAIN EACH EMPOSITION WITHIN JOB-EMPOSITION.

```

SPF Indexes

The following is a section of a subschema that contains SPF indexes. Comparable Integrated Indexes are found in the LRF subschema EMPSS02 listed as EMP-NAME-NDX, JOB-TITLE-NDX, and SKILL-NAME-NDX.

```
ADD SET NAME IS IX-EMP-LNAME
  ORDER IS SORTED
  MODE IS CHAIN
  OWNER IS IXOWNER
    NEXT DBKEY POSITION IS AUTO
  MEMBER IS EMPLOYEE
    NEXT DBKEY POSITION IS AUTO
  OPTIONAL MANUAL
  ASCENDING KEY IS ( EMP-LAST-NAME-0415 )
    DUPLICATES LAST
```

```
ADD SET NAME IS IX-TITLE
  ORDER IS SORTED
  MODE IS CHAIN
  OWNER IS IXOWNER
    NEXT DBKEY POSITION IS AUTO
  MEMBER IS JOB
    NEXT DBKEY POSITION IS AUTO
  OPTIONAL MANUAL
  DESCENDING KEY IS ( TITLE-0440 )
    DUPLICATES NOT ALLOWED
```

```
ADD SET NAME IS IX-SKILL-NAME
  ORDER IS SORTED
  MODE IS CHAIN
  OWNER IS IXOWNER
    NEXT DBKEY POSITION IS AUTO
  MEMBER IS SKILL
    NEXT DBKEY POSITION IS AUTO
  OPTIONAL MANUAL
  ASCENDING KEY IS ( SKILL-NAME-0455 )
    DUPLICATES NOT ALLOWED
```

Customizing the IDMS/DB Environment

The Data Adapter for IDMS/DB provides data retrieval methods and parameters for customizing the environment and optimizing performance. This section explains how your Master File is used to select segments and retrieve data from them. It also discusses:

- How screening conditions (IF criteria) improve I/O efficiency (see *Effects of Screening Conditions on Retrieval* on page 10-65).
- How to work with unique and non-unique segments with short paths (see *Retrieving Short Paths* on page 10-68).
- How you can change retrieval logic by specifying a new root segment (see *File Inversion* on page 10-73).
- How you can join structures together for reporting purposes (see *Joining Master Files* on page 10-75).

The examples in this section reference the EMPFULL Master and Access Files.

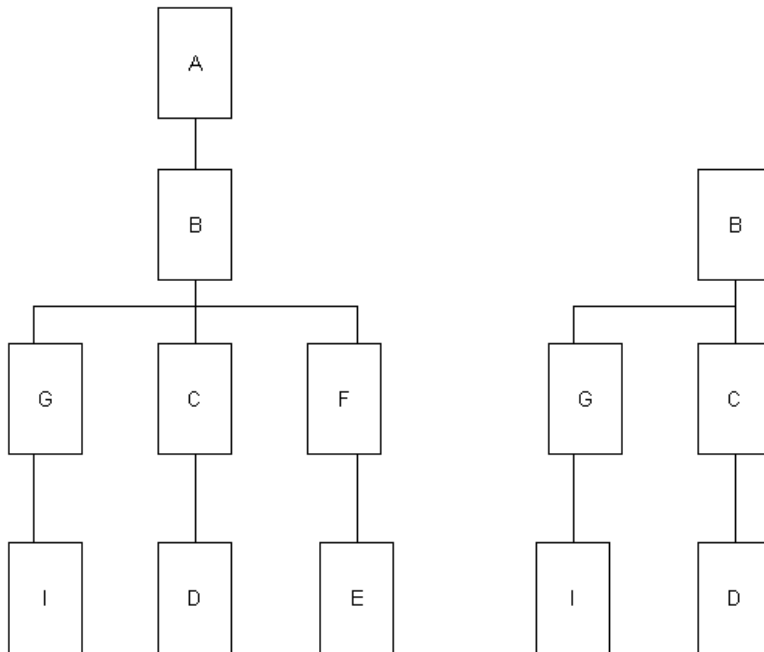
Note: The term *record* refers to a record occurrence; it should not be confused with the IDMS/DB term record-type.

Sections pertaining to unique segments discuss considerations or implications that are true regardless of whether the unique segment is an owner of its parent or is related to its parent through a CALC field, index, or LRF field.

Retrieval Subtree

To retrieve records for a request, a smaller subtree structure is constructed from the structure defined by your Master File. The subtree consists of segments that contain fields named explicitly in the request and those named implicitly by virtual fields or calculated values. The subtree also includes any segments needed to connect these segments.

For example, if a TABLE request needs fields from segments D, G, and I, the subtree shown in the following diagram is constructed:



Master File

Subtree

Segments C and B do not contain fields needed for the request, but they are included in the subtree to connect segment I with segments D and G. Since the segments are descendants of segment B, the entry or root segment of the subtree is segment B. The Master File root segment A is not included, because its fields are not referenced; the records corresponding to segment B can be obtained independently of A. However, if segment B were an OCCURS segment, the IDMS/DB calls would be issued for segment A to obtain segment B's information.

Retrieval Sequence

To determine the order of record retrieval, examine a diagram of the subtree. Use the CHECK FILE command to display the diagram.

Syntax How to Use the Check File Command

```
CHECK FILE filename PICTURE
```

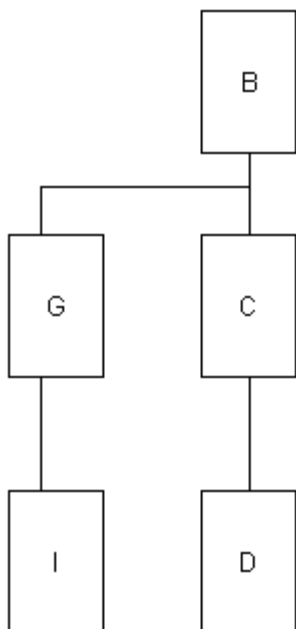
where:

filename

Is the name of the Master File.

The order of retrieval is from top to bottom, left to right. Only the first four fields of each segment are displayed.

For example, in the following figure, the first record in B is retrieved, then the first record in G, and all I records for the first G record. When all the G records and their descendant I records are retrieved, the first record in C is retrieved. Then all the D records for the first C record are retrieved. When all the C records and their descendant D records are retrieved, the next B record is retrieved and the process repeats itself.



Retrieval Sequence With Unique Segments

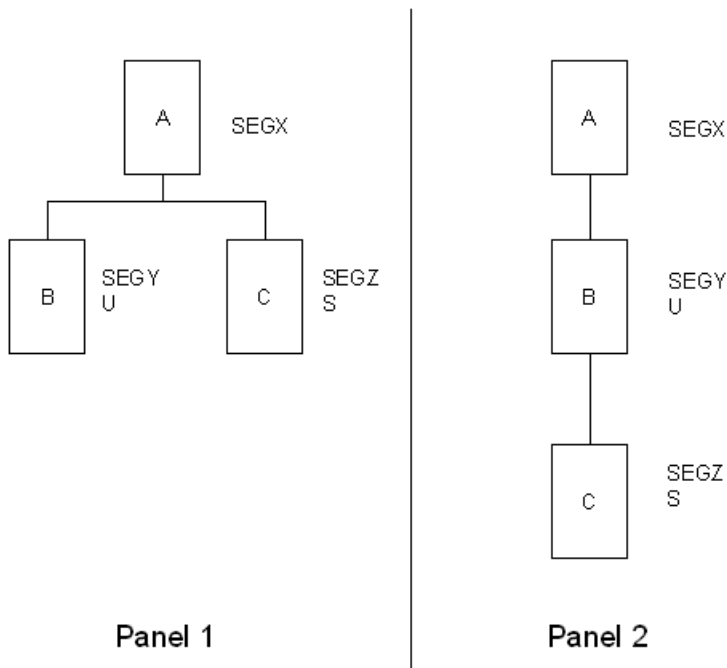
The retrieval sequence for subtrees containing unique segments is still top to bottom, left to right, but the unique segments are treated as extensions of their parents. Records in a unique segment correspond one-to-one with the records in a parent; records in a non-unique segment have a one-to-many correspondence. In cases where the parent segment has unique and non-unique descendants, the unique descendants are always retrieved first regardless of the left-to-right order.

To display a retrieval view of any subtree, use the CHECK FILE command with the RETRIEVE option:

```
CHECK FILE filename PICTURE RETRIEVE
```

A retrieval view shows which sort and IF criteria are valid. For sort phrases (BY or ACROSS), the segment containing the sort field must lie on the same path as the segments with all requested fields. That is, the segment with the BY or ACROSS field must be an ancestor or descendant of the segments containing the required fields.

For example, panel 1 in the following figure shows two descendant segments. The command PRINT B BY C is invalid, because the segments containing fields B and C do not lie on the same path. However, if the segment containing B is a unique segment, the two segments do lie on the same path. Panel 2 shows the retrieval view, and the command PRINT B BY C is valid.



The retrieval sequence for unique segments may also affect the results of COUNT or SUM commands. If a segment is the parent of a unique descendant, there is a one-to-one relationship. A COUNT command like COUNT A AND B returns identical results for each field, because the same record A is counted several times for each record B.

If the parent/descendant relationship is reversed with a non-unique parent, the result for field A is a greater number than the result for field B.

Effects of Screening Conditions on Retrieval

If a record in a segment fails a record selection (IF) test, the corresponding records in descendant segments are not retrieved.

To increase I/O efficiency, place a record selection test at a higher level in the file structure. This restricts the number of records that have to be tested.

Example Using Selection Criteria on Parent Segments

Suppose this request is entered against the structure EMPSS01 (the corresponding Master File is EMPFULL).

```
TABLE FILE EMPFULL
COUNT EMP_ID AS 'NUMBER OF EMPLOYEES'
BY OFFICE_CODE AS 'OFFICE CODE'
IF DEPT_NAME EQ 'PERSONNEL'
END
```

The output is:

```
OFFICE CODE  NUMBER OF EMPLOYEES
-----  -----
002                                4
```

Every time a record in segment DEPT has a value in the DEPT_NAME field not equal to PERSONNEL, the corresponding records in descendant segments EMPLOYEE and OFFICE are ignored, and the next record in segment DEPT is retrieved. In addition, when IF criteria on a lower segment fail, the row is removed from the internal matrix.

Example Using an IF Test on a Descendant Segment

Assume a subtree has four segments:

- DEPT contains department IDs and information.
- EMPLOYEE contains employee names and IDs.
- EMPOSIT contains the positions that the employee has held.
- JOB contains a list of jobs offered by the company.

The following example illustrates how to list all employees who are programmers or analysts:

```
TABLE FILE EMPFULL
PRINT TITLE AS 'TITLE'
BY DEPT_NAME AS 'DEPARTMENT'
BY EMP_LAST_NAME AS 'LAST NAME' IN 25
BY EMP_FIRST_NAME AS 'FIRST NAME'
IF TITLE IS 'PROGRAMMER/ANALYST'
END
```

The output is:

DEPARTMENT	LAST NAME	FIRST NAME	TITLE
INTERNAL SOFTWARE	DOUGH	JANE	PROGRAMMER/ANALYST
	GALLWAY	JAMES	PROGRAMMER/ANALYST
	GRANGER	PERCY	PROGRAMMER/ANALYST
	HEAROWITZ	VLADIMIR	PROGRAMMER/ANALYST
	JENSEN	JULIE	PROGRAMMER/ANALYST
	O'HEARN	KATHERINE	PROGRAMMER/ANALYST

For this request with only one IF test, DML retrieves each DEPT record, each EMPLOYEE record for a given DEPT, each EMPOSIT record for a given EMPLOYEE record, and the JOB record connected to each EMPOSIT. After retrieval, DML determines whether to print the record from the value of the TITLE field.

Example Using Multiple IF Tests to Reduce I/O

The following example produces the same output as *Using an IF Test on a Descendant Segment* on page 10-66. In this company, only the Internal Software department has programmer/analysts working for it. Therefore, adding an additional IF test on the parent segment (DEPT) reduces the number of descendant segments retrieved:

```
TABLE FILE EMPFULL
PRINT TITLE AS 'TITLE'
BY DEPT_NAME AS 'DEPARTMENT'
BY EMP_LAST_NAME AS 'LAST NAME' IN 25
BY EMP_FIRST_NAME AS 'FIRST NAME'
IF DEPT_NAME IS 'INTERNAL SOFTWARE'
IF TITLE IS 'PROGRAMMER/ANALYST'
END
```

Now records are only retrieved and tested when the DEPT_NAME field equals the value INTERNAL SOFTWARE.

Restricting the Number of Records Retrieved

Another method used to increase efficiency is to restrict the number of records retrieved. You can always limit the number of records retrieved from the IDMS/DB data source, and/or the number of records that DML accepts, with READLIMIT and RECORDLIMIT tests. Add one or both of these IF criteria to your request.

Syntax How to Place a Limit on Records Retrieved

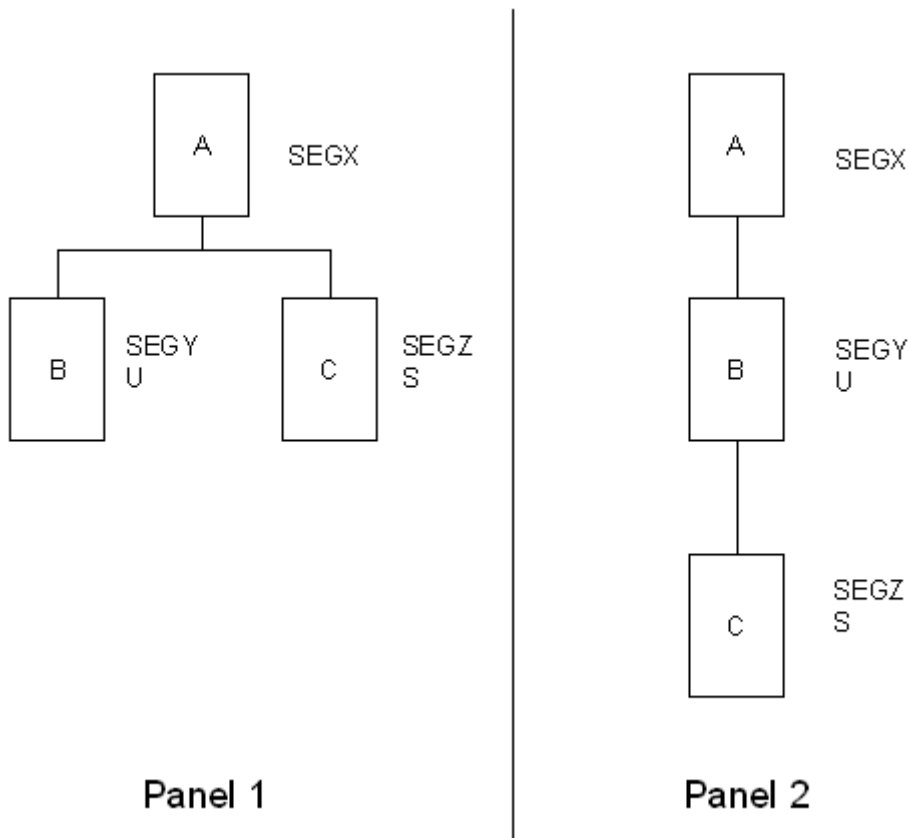
```
IF READLIMIT IS number
IF RECORDLIMIT IS number
```

Tip: These criteria can be assigned to your password as part of file security through the DML DBA facility.

A third method that increases efficiency is file inversion. See *File Inversion* on page 10-73 for more information.

Screening Conditions With Unique Segments

If a record in a unique segment fails a selection test, its parent is rejected and the next record of the parent segment is retrieved. For example, in the following figure, if a record in non-unique segment C fails an IF test, the next record in segment C is retrieved. Only if all C records for a given A fail the test is the A record rejected. When a record in unique segment D fails a test, the parent B record is rejected and the next record in segment B is retrieved. When a record in the entry A segment fails a test, DML retrieves the next A record, even if the entry segment is defined as unique:



Retrieving Short Paths

When a record in a parent segment is retrieved, the corresponding records in the descendant segment are retrieved. If descendant records do not exist, the processing of the parent record depends on whether the descendant segment is unique or non-unique.

Short Paths in Unique Descendants

For a unique descendant segment with a missing record, a temporary record is created to replace the missing record. The temporary record contains fields with default values: blanks for alphanumeric fields and zeros for numeric fields.

For example, an EMPLOYEE segment with the field EMP_NAME has a unique descendant OFFICE segment with the field OFFICE_CITY. The field OFFICE_CITY indicates the location of an employee's office. Gary Smith does not work out of an office location, so he has no OFFICE record. In this situation, all reports that refer to OFFICE_CITY display blank spaces for the entry GARY SMITH.

Short Paths in Non-Unique Descendants

For a non-unique descendant segment with a missing record, the results depend on how the ALL parameter is set:

`SET ALL = {ON|OFF}`

where:

`ON`

The parent record is processed provided that there are no screening conditions on fields in the descendant segment. Missing data is usually indicated on the report by the default NODATA character (.).

`OFF`

The parent instance is rejected and the next parent instance is retrieve. OFF is the default value.

Note: SET ALL = PASS is *not* supported by the adapter.

Example Using SET ALL = OFF

The EMPLOYEE segment has a non-unique descendant segment that contains dental claim records. With SET ALL = OFF, a request that prints employee names and dentist names omits employees that have no dental claims:

```
SET ALL = OFF
TABLE FILE EMPFULL
PRINT EMP_NAME DENTIST_NAME
IF DEPT_NAME EQ 'EXECUTIVE ADMINISTRATION'
                OR 'COMPUTER OPERATIONS'
END
```

The output is:

EMP_NAME		DENTIST_NAME	
-----		-----	
HERBERT	CRANE	DR	PEPPER
HENRIETTA	HENDON	SAL	SARDONICUS

Example Using SET ALL = ON

With SET ALL = ON, a request that prints dentist names and employee names prints everyone's name, even those without dental claims. The report output displays the nodata symbol for the field from the missing descendant segment:

```
SET ALL = ON
TABLE FILE EMPFULL
PRINT EMP_NAME DENTIST_NAME
IF DEPT_NAME EQ 'EXECUTIVE ADMINISTRATION'
           OR 'COMPUTER OPERATIONS'
END
```

The output is:

EMP_NAME		DENTIST_NAME	
-----		-----	
HERBERT	CRANE	DR	PEPPER
JANE	FERNDALE	.	
GEORGE	FONRAD	.	
ROBIN	GARDNER	.	
DOUGLAS	KAHALLY	.	
TERENCE	KLWELLEN	.	
SANDY	KRAAMER	.	
HERBERT	LIPSICH	.	
NANCY	TERNER	.	
HENRIETTA	HENDON	SAL	SARDONICUS
THEMIS	PAPAZEUS	.	
JOHN	RUPEE	.	
ROBBY	WILDER	.	

Example Using SET ALL = ON With Screening Criteria

When a request contains an IF test on the descendant (dental) segment, parent (employee) records are omitted if the descendant segment fails the IF test. Only Herbert Crane has a dental claim for tooth number 99:

```
SET ALL = ON
TABLE FILE EMPFULL
PRINT EMP_NAME DENTIST_NAME
IF DEPT_NAME EQ 'EXECUTIVE ADMINISTRATION'
           OR 'COMPUTER OPERATIONS'
IF TOOTH_NUMBER EQ 99
END
```

The output is:

EMP_NAME		DENTIST_NAME	
-----		-----	
HERBERT	CRANE	DR	PEPPER

Selective ALL Prefix

If the value for SET ALL is OFF, you can still apply the effect of the ON setting to specific segments. To do this, add the ALL prefix to one of the *parent* segment fields in your request. The ALL prefix causes records in that segment to be processed even if they have missing descendants. Like the SET ALL parameter, the processing depends on whether IF criteria exist for the descendant record fields.

Note: The IF criteria provision affects only immediate descendants. In cases where descendants are missing their descendant record occurrences, the parent record occurrence is rejected.

For example, the following request displays people who have no dental claim records; however, it would not display people who have dental claim records and no corresponding dental service records.

```
SET ALL = OFF
TABLE FILE EMPFULL
PRINT ALL.EMP_NAME
DENTIST_NAME SERVICE_DATE
      IF DEPT_NAME EQ 'EXECUTIVE ADMINISTRATION'
                OR 'COMPUTER OPERATIONS'
END
```

The output is:

EMP_NAME		DENTIST_NAME		SERVICE_DATE
-----		-----		-----
HERBERT	CRANE	DR	PEPPER	19800916
HERBERT	CRANE	DR	PEPPER	19800916
JANE	FERNDALE	.		.
GEORGE	FONRAD	.		.
ROBIN	GARDNER	.		.
DOUGLAS	KAHALLY	.		.
TERENCE	KLWELLEN	.		.
SANDY	KRAAMER	.		.
HERBERT	LIPSICH	.		.
NANCY	TERNER	.		.
HENRIETTA	HENDON	SAL	SARDONICUS	19770502
THEMIS	PAPAZEUS	.		.
JOHN	RUPEE	.		.
ROBBY	WILDER	.		.

The ALL prefix is only effective when the SET ALL value is OFF.

File Inversion

When you create a Master File, you create a default representation of a hierarchy. Sometimes, however, you may not want to follow the default route to retrieve records. Two such instances might be when:

1. Your IF criteria screen a segment at the bottom of a subtree.
2. You are processing a multi-path report with IF criteria or sort phrases that are not on a common path.

When these situations occur, you can specify a new entry segment (root) at execution time for a specific request. This process is called file inversion, because the parent/descendant relationships along the path linking the original root and the new root are reversed; other parent/descendant relationships remain unchanged.

Note: File inversions only change the file views; they do not affect the data.

Syntax

How to Invert a File

```
TABLE FILE filename.field
```

where:

field

May be any field in the new root segment.

For example, to invert the EMPFULL file so that the office segment is the new root, specify the field OFFICE_CODE:

```
TABLE FILE EMPFULL.OFFICE_CODE
```

You can also display a diagram of the inverted file with the CHECK FILE command (include the RETRIEVE option for a subtree diagram):

```
CHECK FILE filename.fieldname PICTURE [RETRIEVE]
```

You cannot invert a Master File if:

- The path linking the old and new roots passes through segments that have a CALC- or index-based relationship.
- The GETOWN parameter in the ACCESS file for a set-based relationship is set to N.
- It is an LRF Master File.

File inversion is a simple solution to two common problems:

- Denied access because the segment is on the wrong sort path.
- Denied access because the field named in an IF test is not on the root path.

Example Using File Inversion to Solve Sort Path Problems

In addition to solving the sort path problem, file inversion can improve I/O efficiency which, in turn, minimizes production costs.

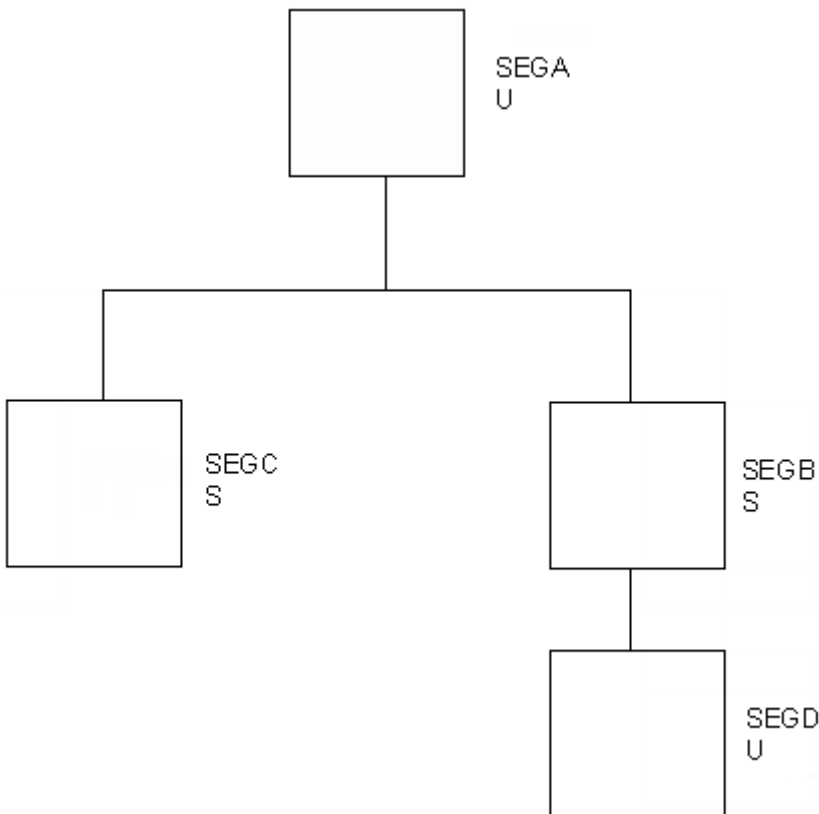
Consider this request:

```
TABLE FILE EMPFULL  
LIST SKILL_LEVEL BY SALARY_GRADE  
END
```

In panel 1 of the following figure, an error occurs because segments C and B are not on the same path. Therefore, you must use an inverted view:

```
TABLE FILE EMPFULL.SALARY_GRADE  
LIST SKILL_LEVEL BY SALARY_GRADE  
END
```

In the inverted view (panel 2), segment C is a descendant of segment B. Using this inverted view, the request can be executed.



As this request is executed, record occurrences multiply. Every record of segment C is paired with every record in segment B. If, for example, A had two B descendants and four C descendants, the report would contain eight lines of output. This effect is advantageous when it is necessary to pair every record associated with one linkpath to a record associated with another linkpath. Record pairing may produce undesirable results when the inverted segments are not directly related to each other.

If you use file inversion in conjunction with MISSING=ON, you may access orphan record occurrences that could not be accessed with the default Master File. An orphan record occurrence is one that has no parent record connection. Due to the network structure of IDMS/DB, any hierarchical view may contain orphans. IDMS/DB set connection options OA, OM, or MM indicate the possibility of orphans. Inversion enables the adapter to reconstruct the IDMS/DB relationships so that these orphans can be retrieved.

Joining Master Files

You can join the Master Files describing any of these data sources to that of your IDMS/DB data source:

- Other IDMS/DB data sources (SUFFIX=IDMSR)
- VSAM or ISAM or QSAM
- SQL or UDB (DB2[®])
- DOS/DL1 or IMS[™]
- CA-Datcom[®]/DB
- MODEL 204[®]
- Fixed-format sequential

A JOIN structure is implemented by matching one field that is common to both data sources. The fields on the IDMS/DB target file can be:

- An IDMS/DB CALC field on a network record-type.
- An indexed field (FIELDTYPE=I) on a network record-type.
- A field on an LRF record.

The fields on the host file can be:

- A virtual field located in a host Master File or created as a separate command.
- Any field.

In the Master File, the names of common fields can differ, but their field formats (ACTUAL and USAGE) must be the same.

Syntax **How to Join Two Data Sources**

```
JOIN field1 [WITH rfield] IN hostfile [TAG tag1]  
TO [ALL] field2 IN crfile [TAG tag2] [AS name]  
[END]
```

where:

field1, field2

Are the fields common to both Master Files.

WITH *rfield1*

Use only if *field1* is a virtual field; assigns a logical home with a real field in the host file.

hostfile

Is the host Master File.

TAG *tag1*

Is a tag name of up to eight characters (usually the name of the Master File), which is used as a unique qualifier for fields and aliases in the host file.

The tag name for the host file must be the same in all the JOIN commands of a joined structure.

ALL

Use if non-unique relationships exist in the target file.

crfile

Is the target or cross-referenced Master File.

TAG *tag2*

Is a tag name of up to eight characters (usually the name of the Master File), which is used as a unique qualifier for fields and aliases in the cross-referenced file. In a recursive joined structure, if no tag name is provided, all field names and aliases are prefixed with the first four characters of the join name.

AS *name*

Assigns a name to the JOIN structure. You must assign a unique name to a join structure if:

- You want to ensure that a subsequent JOIN command will not overwrite it.
- You want to clear it selectively later.
- The structure is recursive, and you do not specify tag names.

END

Required when the JOIN command is longer than one line; terminates the command.

To join more than two files as a single structure, indicate the common fields as follows:

```
JOIN field1 IN file1 TO field2 IN file2 AS name1
JOIN field3 IN file1 TO field4 IN file3 AS name2
```

Reference Usage Notes for the JOIN Command

- Up to 16 joins may be active in one session.
- For a DML target file, field2 must be indexed (FIELDTYPE=I).
- If you intend to use a virtual field as *field1*, specify its field name in the JOIN command and then issue its DEFINE command. Any DEFINE commands issued prior to the JOIN are cleared.
- If you know that the target file is unique, omit the ALL in the JOIN command; omitting ALL reduces I/O overhead.
- To display the JOIN structure, use the CHECK FILE command and specify the name of the host file.

Syntax How to List JOIN Structures

To list your JOIN structures, enter:

```
? JOIN
```

Syntax How to Clear JOIN Structures

To clear a specific JOIN structure, specify the name that you assigned to the join:

```
JOIN CLEAR name
```

Syntax How to Clear All JOIN Structures

To clear all structures, use an asterisk (*) instead of a join name:

```
JOIN CLEAR *
```

Example Reporting From a Joined Structure

This example joins the DML data source JOBFIL to the IDMS/DB EMPFULL data source based on job codes. First the JOBCODE field in JOBFIL is edited to make it compatible with the JOB_ID field in EMPFULL. The JOIN command is issued prior to the DEFINE. If the DEFINE were issued first, it would be cleared by the JOIN command:

```
JOIN JOBID WITH JOBCODE IN JOBFIL TO
ALL JOB_ID IN EMPFULL AS J1
END
DEFINE FILE JOBFIL
JCODE/A2 = IF JOBCODE LIKE 'A__' THEN '10' ELSE '20';
JOBID/A4 = JCODE|EDIT(JOBCODE, '$99');
END
TABLE FILE JOBFIL
SUM EMP_NAME IN 25 TITLE
BY DEPT_NAME
END
```

The output is:

DEPT_NAME	EMP_NAME	TITLE
ACCOUNTING AND PAYROLL PERSONNEL	RUPERT ELEANOR	JENSON PEOPLES MGR ACCTNG/PAYROLL MGR PERSONNEL

Data Adapter for IDMS/DB Navigation

The Data Adapter for IDMS/DB retrieves the necessary records that fulfill a request. It uses information from the following sources to choose the most appropriate and efficient retrieval method:

- File Retrieval
- Record Retrieval Internals

IDMS/DB File Retrieval

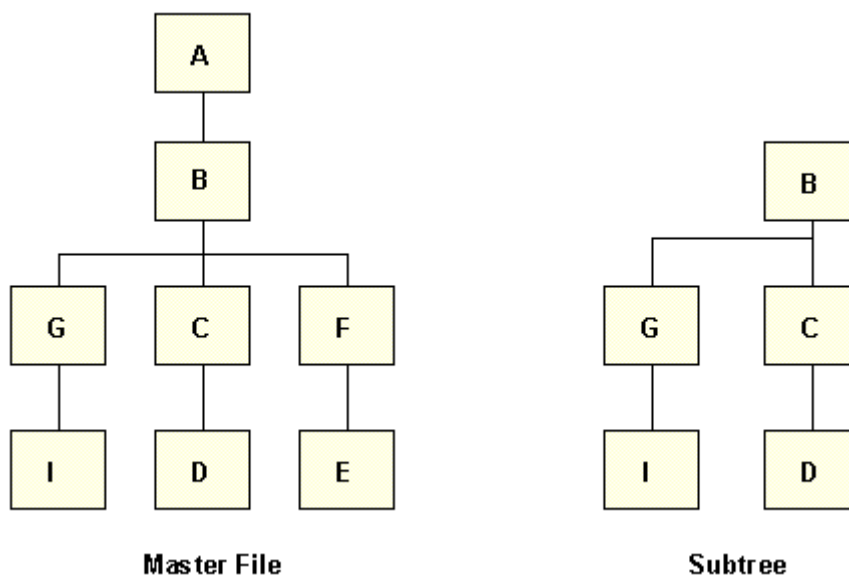
The server uses the Master File to select segments and retrieve data from them.

Note: Sections pertaining to unique segments discuss rules that apply regardless of whether the unique segment is the owner of its parent or is related to its parent through a CALC field, index, or LRF field.

Retrieval Subtree

To retrieve records for a request, the server first constructs a smaller subtree structure from the structure defined by the Master File. The subtree consists of segments that contain fields named explicitly in the request and those named implicitly by DBA DEFINE or COMPUTE statements. The subtree also includes any segments needed to connect these segments.

For example, if an SQL request needs fields from segments D, G, and I, the server constructs the subtree shown below. Segments C and B do not contain fields needed for the request, but they are included in the subtree to connect segment I with segments D and G. Since the segments are descendants of segment B, the entry or root segment of the subtree is segment B. The Master File root segment A is not included, because its fields are not referenced; the records corresponding to segment B can be obtained independently of A. However, if segment B were an OCCURS segment, the IDMS/DB calls would be issued for segment A to obtain B's information.

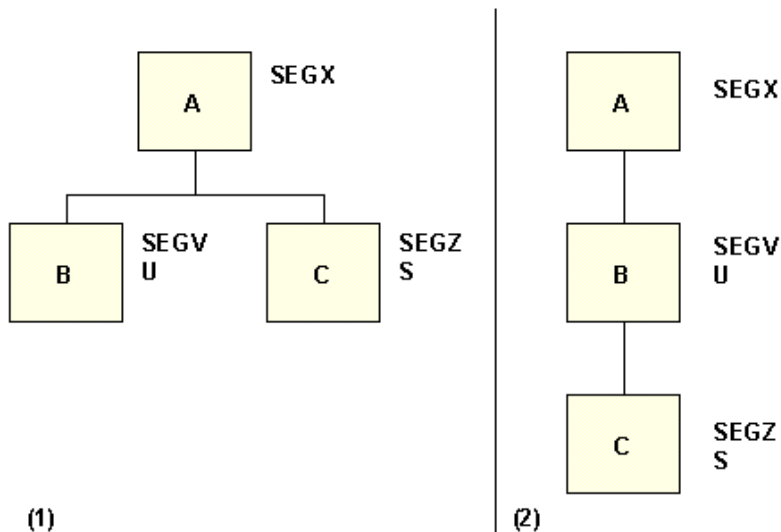


Retrieval Sequence With Unique Segments

The retrieval sequence for subtrees containing unique segments is still top to bottom, left to right, but the unique segments are treated as extensions of their parents. Records in a unique segment correspond one-to-one with the records in a parent; records in a non-unique segment have a one-to-many correspondence. In cases where the parent segment has unique and non-unique descendants, the unique descendants are always retrieved first regardless of the left-to-right order.

A retrieval view shows which sort and WHERE clauses are valid. For sort statements (BY or ACROSS), the segment containing the sortfield must lie on the same path as the segments with all requested fields. That is, the segment with the BY or ACROSS field must be an ancestor or descendant of the segments containing the required fields.

For instance, panel 1 of the following graphic shows two descendant segments. The statement `SELECT B ORDER BY C` is invalid, because the segments containing fields B and C do not lie on the same path. However, if the segment containing B is a unique segment, the two segments do lie on the same path. Panel 2 shows the retrieval view; the statement `SELECT B ORDER BY C` is valid.



The retrieval sequence for unique segments may also affect the results of an SQL statement that contains `COUNT` or `SUM`. If a segment is the parent of a unique descendant, there is a one-to-one relationship. A `COUNT` statement, such as `COUNT A AND B`, returns identical results for each field, because the same record A is counted several times for each record B. If the parent/descendant relationship is reversed with a non-unique parent, the result for field A is a greater number than the result for field B.

Screening Conditions

If a record in a segment fails a WHERE condition, the server does not retrieve the corresponding records in descendant segments. Suppose this request is entered against the structure EMPSS01 (corresponding Master File is EMPFULL).

```
SELECT EMP_ID, OFF_CODE
FROM EMPFULL
WHERE DEPT_NAME = 'PERSONNEL'
ORDER BY OFF_CODE
```

Every time a record in segment DEPT has a value in the DEPT_NAME field not equal to PERSONNEL, the server ignores the corresponding records in descendant segments EMPLOYEE and OFFICE, and retrieves the next record in segment DEPT. In addition, when a WHERE clause on a lower segment fails, the row is removed from the server answer set.

To increase I/O efficiency, place the WHERE clauses at a higher level in the file structure. This restricts the number of records the server has to test. The example below shows the benefits of two WHERE clauses versus one.

Assume a subtree has four segments:

- DEPT contains department IDs and information.
- EMPLOYEE contains employee names and IDs.
- EMPOSIT contains the positions that the employee has held.
- JOB contains a list of jobs offered by the company.

To list all employees who are programmer/analysts:

```
SELECT TITLE, DEPT, LAST_NAME, FIRST_NAME
FROM EMPFULL
WHERE TITLE = 'PROGRAMMER/ANALYST'
ORDER BY DEPT, LAST_NAME, FIRST_NAME
```

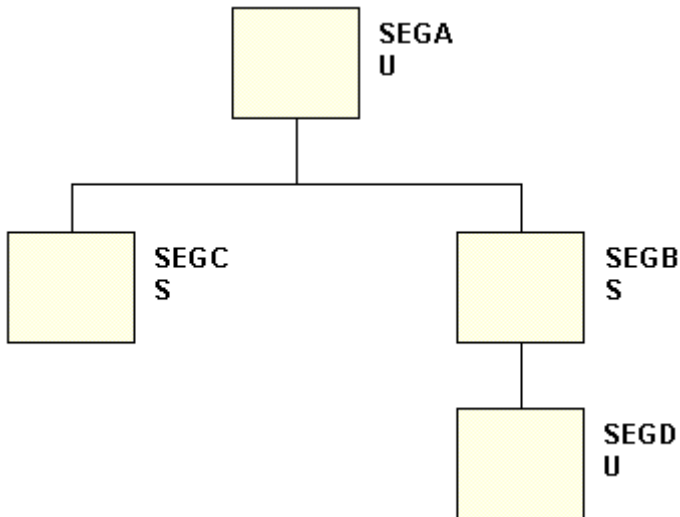
For this request with only one WHERE clause, the server retrieves each DEPT record, each EMPLOYEE record for a given DEPT, each EMPOSIT record for a given EMPLOYEE record, and the JOB record connected to each EMPOSIT. After retrieval, the server determines whether to include in the answer set the record from the value of the TITLE field. To make retrieval more efficient, add another WHERE clause on a segment higher in the structure. In this company, only the Internal Software department has programmer/analysts working for it:

```
SELECT TITLE, DEPT, LAST_NAME, FIRST_NAME
FROM EMPFULL
WHERE DEPT_NAME = 'INTERNAL SOFTWARE'
WHERE TITLE = 'PROGRAMMER/ANALYST'
```

Now the server retrieves and tests records only when the DEPT_NAME field equals the value INTERNAL SOFTWARE.

Screening Conditions With Unique Segments

If a record in a unique segment fails a WHERE test, the server rejects its parent and retrieves the next record of the parent segment. For example, in the following graphic, if a record in non-unique segment C fails a WHERE clause, the server retrieves the next record in segment C. Only if all C records for a given A fail the test is the A record rejected. When a record in unique segment D fails a test, the server rejects the parent B record and retrieves the next record in segment B. When a record in the entry A segment fails a test, the server retrieves the next A record, even if the entry segment is defined as unique.



Short Paths

When the server retrieves a record in a parent segment, it retrieves the corresponding records in the descendant segment. If descendant records do not exist, the processing of the parent record and whether it is included in an answer set depends on whether the descendant segment is unique or non-unique.

Short Paths in Unique Descendants

For a unique descendant with a missing record, the server creates a temporary record to replace the missing record. The temporary record contains fields with default values: blanks for alphanumeric fields and zeroes for numeric fields.

For example, an EMPLOYEE segment with the field EMP_NAME has a unique descendant OFFICE segment with the field OFF_CITY. The field OFF_CITY indicates the location of an employee's office. Gary Smith does not work out of an office location; so, he has no OFFICE record. In this situation, all requests that refer to OFF_CITY display blank spaces for the entry GARY SMITH.

Short Paths in Non-Unique Descendants

For a non-unique descendant segment with a missing record, the server rejects the parent instance and retrieves the next parent instance.

CA-IDMS/DB Record Retrieval Internals

To obtain all of the necessary records to fulfill a request, the Data Adapter for IDMS/DB navigates the IDMS/DB database using DML or LRF commands. The adapter automatically generates DML or LRF commands based on information from the Master File, Access File, and your request for the most appropriate and efficient IDMS/DB retrieval method.

There are three kinds of IDMS/DB access:

- Entry Segment Retrieval of Network Records.
- Descendant Segment Retrieval of Network Records.
- LRF Record Retrieval (LR and ASF).

Subsequent sections discuss the navigational strategies used for each kind of access.

Entry Segment Retrieval of Network Records

The server constructs a retrieval subtree based on the Master and Access Files and your request. The root of this subtree is called the entry segment, because the server begins its retrieval search of the database at that point. The actual IDMS/DB retrieval calls used on the entry segment depend on the entry segment's Access File information and any WHERE clauses. To perform the most efficient record retrieval on the entry segment, the data adapter chooses one of the following techniques:

- Retrieval by Database Key
- Retrieval by CALC Field
- Retrieval by Index
- SEQFIELD Parameter
- Retrieval by Area Sweep

These techniques are listed in descending order of efficiency. The idea behind selection logic is to perform as many WHERE clauses as possible at the IDMS/DB level. This minimizes the actual I/O operations required to access the necessary data. Area sweeps are the least desirable retrieval technique, because they read through every record type in the named area, including record types that correspond to other segments, and return every entry segment record to the server. At this point, the server selects those records that satisfy the request's test criteria and discards the rest.

Retrieval by Database Key

The IDMS/DB database key method of retrieval takes precedence over the other methods because it is the most efficient. This method depends on the existence of two conditions:

- A field that corresponds to the IDMS/DB database key (ALIAS=DBKEY) for the entry segment.
- An equality test in the request on the DBKEY field.

The equality test sets the field name of the DBKEY from the entry segment in the Master File equal to a specified numeric value(s):

```
WHERE field = 'value1'
```

Retrieval by CALC Field

If there is no WHERE clause on the DBKEY for the entry segment, the second choice is CALC access. Retrieval through the CALC key, while not as efficient as DBKEY access, takes precedence over an index or area sweep retrieval.

The CALC retrieval method depends on the existence of two conditions:

- The entry segment must contain a CALC keyfield as specified in the Access File.
- A fully-qualified equality test in the request on the CALC field.

The WHERE clause sets the field name of the CALC key for the segment equal to fully qualified values:

```
WHERE field = 'value1'
```

For each value specified in the WHERE clause, the data adapter calls IDMS/DB with the following DML command:

```
OBTAIN CALC record
```

If the Access File indicates duplicate records (CLCDUP=Y), each DML call is followed by subsequent calls:

```
OBTAIN DUPLICATE record
```

This ensures that all appropriate records are obtained to satisfy the request.

Retrieval by Index

If there is no DBKEY or CALC key test criteria in the request, the data adapter selects the index retrieval method.

Note: An index field must be defined with the FIELDTYPE=I attribute in the Master File and a corresponding index declaration must exist in the Access File.

Index retrieval is performed using:

- WHERE clauses for an indexed field or GROUP in the entry segment.
- The optional parameter SEQFIELD in the Access File segment declaration.

The first method requires at least one WHERE clause that specifies the field name of the index field. The following WHERE clause invokes index-based retrieval:

- Fully-qualified.

```
WHERE = 'PROGRAMMER/ANALYST'
```

- Partially-qualified (generic); also called masking. This applies to alphanumeric fields only.

```
WHERE TITLE LIKE 'PROGRAMMER%'
```

- Specified as a range of values.

```
WHERE TITLE BETWEEN 'PROGRAMMER' AND 'WORD PROCESSOR'
```

When two or more WHERE clauses in a request qualify an index on the entry-level segment, fully-qualified retrieval takes precedence over generic; generic retrieval takes precedence over range. If two WHERE clauses are the same type, the index in the first WHERE clause is used. All types of WHERE clauses are also supported for indices that allow duplicate values.

If the test criteria indicates index retrieval, the data adapter issues this DML command to IDMS/DB:

```
OBTAIN FIRST record WITHIN setname USING value
```

In this command, setname is the name of the index set specified in the Access File.

Then, if the Access File indicates duplicate records, the data adapter issues this DML command for index sets:

```
OBTAIN NEXT record WITHIN setname
```

The above OBTAIN NEXT call is issued until all the duplicate records are retrieved. This same NEXT call is also issued if your request contains generic or range WHERE clauses. It is performed once for every value or range specified in the WHERE clause.

Note: If the IXORD parameter is improperly specified in the Access File, a range WHERE clause may erroneously produce an answer set with one or no records.

SEQFIELD Parameter

The second method of index retrieval does not require WHERE clauses, and yet prohibits area sweeps on entry segments. To use this method, add the optional SEQFIELD parameter to the Access File and specify the name of the indexed field as the value of SEQFIELD.

When your request does not contain a WHERE (for DBKEY, CALC key, or another index) and a SEQFIELD is specified for the entry segment, the data adapter issues this DML command to IDMS/DB:

```
OBTAIN FIRST record WITHIN setname
```

In this command, *setname* is the IDMS/DB set name of the indexed field (IXSET parameter) from the Access File.

The data adapter then issues the next command until all records connected to the index set are retrieved:

```
OBTAIN NEXT record WITHIN setname
```

If no sort criteria (ORDER BY) is specified in the request, the answer set is produced in ascending or descending index set order.

The SEQFIELD method is recommended for indexed segments in large IDMS/DB databases where only a small percentage of record occurrences in a given area are the record types defined by the segments. In such cases, IDMS/DB resource utilization can be greatly reduced through the use of this parameter.

Note: If the index set connection is not mandatory/automatic (MA), some of the records in this record type may not be accessed if it is the entry segment. In this situation, only records that are members of the index set are supplied to the server. If this retrieval result is undesirable, you should omit the SEQFIELD parameter.

Retrieval by Area Sweep

An area sweep is the least efficient method of entry-level retrieval, because it reads through every record in an IDMS/DB area to return records of a given record type. Despite its inefficiency, an area sweep is sometimes the only method available for retrieval.

The data adapter performs an area sweep if one of the following occurs:

- No equality WHERE on the DBKEY for the root exists.
- No equality WHERE on the CALC key for the root exists.
- No equality or range WHERE on an indexed field exists.
- No SEQFIELD parameter is specified.

If one of the above situations occurs, the data adapter issues this DML command to IDMS/DB:

```
OBTAIN FIRST record WITHIN areaname
```

In this command, *areaname* is the name of the IDMS/DB database area specified in the Access File. Next, the data adapter continues to issue this command until all the records are obtained:

```
OBTAIN NEXT record WITHIN areaname
```

Descendant Segment Retrieval of Network Records

To retrieve records from a descendant segment, the data adapter's navigational strategy depends on Access File parameters and the SEGTYPE parameter in the Master File. In a few cases, the WHERE clauses in a request also affect the strategy.

In general, the ACCESS parameter in the Access File determines retrieval strategy, because it indicates how parent/descendant relationships are implemented. There is a retrieval strategy for each kind of relationship:

- Set-Based Retrieval
- CALC-Based Retrieval
- Index-Based Retrieval
- LRF Record Retrieval
- Overriding DBNAME and DICTNAME

Set-Based Retrieval

For a set relationship (ACCESS=SET), the IDMS/DB set is searched starting with the owner to obtain related member record(s). Set relationships are physical ones, implemented by set pointer chains. The SEGTYPE parameter in the Master File indicates whether the descendant segment is unique or non-unique.

If the descendant segment is non-unique (SEGTYPE=S), it represents a member record type. For non-unique descendants, the data adapter issues this command:

```
OBTAIN NEXT record WITHIN setname
```

This command is repeated until IDMS/DB indicates that the end of the set is reached. Then the data adapter obtains records of other descendant segments for the same parent segment. If no other descendant segments exist, the next parent record is retrieved.

If the descendant segment is unique (SEGTYPE=U), it represents an owner record type. For unique descendant segments, the data adapter issues this command:

```
OBTAIN OWNER WITHIN setname
```

The command is issued once, since there is one owner per set. The data adapter continues to retrieve descendant records for the same parent or retrieves the next parent record.

The KEYFLD and MULTMBR parameters in the Access File also affect retrieval for certain requests. For a sorted set, the KEYFLD parameter specifies that the set is ordered by a specified field. When the request references a field from the parent segment and has a WHERE (=, BETWEEN, >, >=, <, <=) on the sortfield, the data adapter sends this command to IDMS/DB:

```
OBTAIN record WITHIN setname USING value
```

If there are duplicate records (SETDUP=Y), the data adapter issues this command:

```
OBTAIN NEXT record WITHIN setname
```

When the value of the sortfield changes beyond the specified range, retrieval for that segment stops. I/O operations are minimized when the sortfield value is supplied in the OBTAIN command.

Note: The means of implementing the sorted set—the traditional method or using IDMS/DB Integrated Indices—is transparent to the data adapter.

The MULTMBR parameter indicates an IDMS/DB multi-member set. When it is specified, the data adapter searches for other member record types (segments) in the Access File with the same setname. As a result, all necessary IDMS/DB areas are activated.

CALC-Based Retrieval

CALC-based relationships (ACCESS=CLC) are performed with embedded cross-references. A field in the parent segment corresponds to the CALC field in its descendant segment. The data adapter uses the value from the parent's field and performs entry-level IDMS/DB retrieval. The process of retrieving records from a descendant segment is similar to that of an entry segment. The difference is that the value supplied by the parent segment acts as the WHERE clause as if it were an explicit WHERE.

After the data adapter retrieves the host field value (KEYFLD=value) from the parent segment, it calls IDMS/DB:

```
OBTAIN CALC record
```

Then, if the descendant segment is non-unique (SEGTYPE=S), the data adapter issues this command until all of the appropriate records are obtained to satisfy the request:

```
OBTAIN DUPLICATE record
```

Note: If the CLCDUP parameter does not correspond to the SEGTYPE parameter, message EDA919 displays.

A descendant segment with a CALC-based relationship (ACCESS=CLC) may act as a parent and be related to its descendants using set-, CALC-, or index-based relationships.

Index-Based Retrieval

Like CALC-based relationships, index-based relationships (ACCESS=IX) also use embedded cross-references. In index-based relationships, the field in a descendant segment represents an index on the IDMS/DB record type. The index can be either a Sequential Processing Facility (SPF) index or an Integrated Index. The data adapter uses the value from the parent segment and performs entry-level IDMS/DB retrieval by searching the index set. The process of retrieving records from a descendant segment is similar to that of an entry segment. The difference is that the value supplied by the parent segment acts as the WHERE clause, as if it were an explicit WHERE.

After the data adapter retrieves the host field value (KEYFLD=value) from the parent segment, it calls IDMS/DB:

```
OBTAIN FIRST record WITHIN setname USING value
```

Then, if the descendant segment is non-unique (SEGTYPE=S), the data adapter issues the following command until all indexed records with the host value are retrieved:

```
OBTAIN NEXT record WITHIN setname
```

Note: If the IXDUP parameter does not correspond to the SEGTYPE parameter, message EDA919 displays.

Only a descendant segment with an Integrated Index may act as a parent and be related to its descendants using set-, CALC-, or index-based relationships.

LRF Record Retrieval

To retrieve LR and ASF records, the data adapter sends LRF calls to an access module IDMS/DB which invokes Logical Record Facility program.

LRF-based records are retrieved when the Access File specifies MODE=LR in the subschema declaration and the data adapter constructs an LR call to IDMS/DB with explicit or implicit WHERE clauses from the request. The LRF processes the request as generated by the data adapter, selects the appropriate LR path, and constructs each flat view using the full set of WHERE clauses. The process is highly efficient in terms of I/O; only those records which pass the WHERE clause are passed back to the data adapter from IDMS/DB.

The retrieval process for LRF records is identical to that of network record types, but the Logical Record Facility maintains its own navigational information for the database, selects the retrieval strategy most appropriate for a given request, and maintains its own set of occurrences.

When the subschema mode is Logical Record (MODE=LR), the data adapter analyzes the request and creates an LRF command to be sent to the Logical Record Facility. The LRF command has two formats. The first format is for an entry segment without WHERE clauses:

```
OBTAIN NEXT record
```

The second format is for an entry with WHERE clauses or for a descendant segment:

```
OBTAIN NEXT record WHERE expression1 [AND expressionN]
```

As indicated by the brackets, this command is also used to pass compound WHERE clauses to IDMS/DB.

IDMS/DB processes the data adapter's call and returns a record if two conditions are met:

- There is a SELECT path in the LR path-group that can process the particular request. (**Note:** For entry segments without WHERE clauses, there must be a null SELECT clause defined for the logical record.) If there is no available SELECT path, IDMS/DB returns message 2041, and the data adapter terminates its processing with an EDA949 message.
- There is a complete LRF record created by the path-group logic. If the selected path-group can return partial records, the data adapter processes returned records until the first partial record is returned and the LR status field is not LR-FOUND or LR-NOT-FOUND. When the data adapter encounters any other status in the LR status field, it aborts the record retrieval process and returns messages EDA967 and EDA949. Logical records that allow for retrieval of partial records should not be used with the data adapter.

The data adapter continues to call IDMS/DB for LRF records that correspond to a segment until IDMS/DB returns LR-NOT-FOUND in the LR status field. Then the data adapter retrieves records for other segments, or it terminates retrieval and the answer set is produced.

Overriding DBNAME and DICTNAME

You can dynamically override the DBNAME and DICTNAME parameters within the Access File. This allows users to specify the DBNAME and DICTNAME for all IDMS/DB Master File and Access File pairs during a session by using a SET command, eliminating the need to modify each Access File manually.

To override the DBNAME and DICTNAME, issue the SET commands outlined below. Once the SET commands are issued, the DBNAME and DICTNAME specified will override the same keywords in all Access Files during your session until you end your session or set the DBNAME and DICTNAME to default.

If no SET command is issued, the default behavior is followed.

Syntax How to Set DBNAME and DICTNAME

```
ENGINE IDMSR SET DBNAME dbname
```

```
ENGINE IDMSR SET DICTNAME dictname
```

where:

dbname

Is the IDMS/DB database name that you want to access.

dictname

Is the IDMS/DB dictionary name that you want to access.

Note: These set commands can be included in any supported server profile.

Syntax How to Display the Current Settings

To display the settings that are currently in effect, issue the following command from a client application or from a remote procedure:

```
EX EDAEXEC 'ENGINE IDMSR SET ?'
```

Syntax How to Revert to Original Settings

To revert to original settings in the Access File, issue the following commands in a remote procedure:

```
ENGINE IDMSR SET DBNAME DEFAULT
ENGINE IDMSR SET DICTNAME DEFAULT
```

Syntax How to Override the DBNAME and DICTNAME in All IDMS/DB Access Files With SYSDIRL

```
ENGINE IDMSR SET DBNAME SYSDIRL
ENGINE IDMSR SET DICTNAME SYSDIRL
```

Tracing the Data Adapter for IDMS/DB

From the Web Console main screen, select *Diagnostics* and click *Enable Traces*. The default traces include the Data Adapter for IDMS/DB trace information.

CHAPTER 11

Getting Started in CA-IDMS/SQL

Topics:

- Preparing the IDMS/SQL Environment
- Configuring the Data Adapter for IDMS/SQL
- Managing IDMS/SQL Metadata
- Customizing the IDMS/SQL Environment

The Data Adapter for CA- IDMS/SQL allows applications to access IDMS/SQL data sources. The adapter converts application requests into native IDMS/SQL statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

In order to use the Data Adapter for IDMS/SQL, you must configure and map data to its corresponding server counterparts. Check with your Server Administrator for further information.

Note: For the remainder of this manual, the name IDMS/SQL refers to CA-IDMS/SQL.

Preparing the IDMS/SQL Environment

Prior to configuring the Data Adapter for IDMS/SQL using the Web Console, it is necessary to edit the ISTART JCL used to initiate the server. The changes required are documented as follows:

- Allocate the IDMS.LOADLIB and IDMS.DBA.LOADLIB libraries to the server's STEPLIB ddname allocation.
- Allocate ddname SYSCTL to the IDMS.SYSCTL data set.
- Allocate ddname SYSIDMS to the IDMS.SYSIDMS data set.

Configuring the Data Adapter for IDMS/SQL

Configure the adapter using the Web Console Adapter Add screen and click *Configure*.

Overriding the Default Connection

An SQL session is a connection between the application and the IDMS/SQL data source. It begins when the application connects to a dictionary. You use the CONNECT command to override the IDMS/SQL default (automatic) connection. The length of time an SQL session stays in effect depends on whether the connection began automatically or a CONNECT command was issued. If the CONNECT command was issued, the SQL session is in effect until a COMMIT RELEASE, ROLLBACK RELEASE, or RELEASE command is executed. All of these commands may be executed within the IDMS/SQL session using SQL Passthru. Refer to the appropriate IDMS/DB documentation for more information regarding SQL sessions.

Syntax

How to Override Default Connections With the CONNECT Command

Issue the following syntax within a stored procedure

```
ENGINE [SQLIDMS] CONNECT TO dictionary
```

where:

SQLIDMS

Indicates the IDMS/SQL data source. You can omit this value if you previously issued the SET SQLENGINE command.

dictionary

Is the data source (dictionary) to start the IDMS/SQL session. The default is the dictionary in effect for the user session. This default is set outside of the server session, for example, with a SYSIDMS DICTNAME parameter. Please refer to the appropriate CA-IDMS/DB documentation for a complete description.

Other Session Commands

Other IDMS/SQL commands that affect the IDMS/SQL session can be executed explicitly.

To issue IDMS/SQL session commands such as COMMIT, COMMIT RELEASE, ROLLBACK, ROLLBACK RELEASE, and COMMIT CONTINUE, the syntax is:

```
ENGINE [SQLIDMS] COMMIT RELEASE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax How to Control the Connection Scope

```
ENGINE [SQLIDMS] SET AUTODISCONNECT ON {COMMIT|FIN}
```

where:

SQLIDMS

Indicates the IDMS/SQL data source. You can omit this parameter value if you previously issued the SET SQLENGINE command.

COMMIT

Disconnects when a COMMIT or ROLLBACK is issued as a native SQL command. This setting frees the thread of execution for use by other users. The disadvantage is the cost of repeatedly connecting and acquiring a thread. Threads, once released, may not be available when needed, so you may experience delays while your request waits for a thread.

FIN

Disconnects automatically only after the server session has been terminated.

Depending on how often the event occurs, the AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server; it is related to the operating system and the data source.

Managing IDMS/SQL Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the IDMS/SQL data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each IDMS/SQL table or view that is accessible from the server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

Syntax

How to Create a Synonym Manually

```
CREATE SYNONYM appname/synonym FOR datasource DBMS SQLIDMS  
END
```

where:

appname

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used. If your server is not APP enabled, this application name must not be used.

synonym

Is an alias for the data source (it has a maximum of 64 characters).

datasource

Is the SCHEMA.TABLENAME.

SQLIDMS

Indicates the Data Adapter for IDMS/SQL.

END

Indicates the end of the command and is required on a separate line in the stored procedure.

Note:

- CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.
- CREATE SYNONYM creates a Master File and Access File which represents the server metadata.

Example Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.NF29004 DBMS SQLIDMS
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION ,SUFFIX=SQLIDMS,$
SEGNAME=SEG1_4 ,SEGTYPE=S0 ,$
FIELD=DIVISION4 ,DIVISION4 ,I9 ,I4 ,MISSING=OFF ,$
FIELD=DIVISION_NA4,DIVISION_NA4 ,A25 ,A25 ,MISSING=ON ,$
FIELD=DIVISION_HE4,DIVISION_HE4 ,I9 ,I4 ,MISSING=ON ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4, TABLENAME=EDAQA.NF29004,
CONNECTION=IDMS901, KEYS=1, WRITE=YES, $
```

Master Files

The following describes the three types of Master File declarations:

Declaration Type	Description
File	Names the file and describes the type of data source.
Segment	Identifies a table, file, view, or segment.
Field	Describes the columns of the table or view.

Each declaration must begin on a separate line. A declaration consists of attribute-value pairs separated by commas. A declaration can span as many lines as necessary, as long as no single keyword-value pair spans two lines.

Do not use system or reserved words as names for files, segments, fields, or aliases. Specifying a reserved word generates syntax errors.

Syntax How to Declare File Attributes

Each Master File begins with a file declaration. The file declaration has two attributes:

FILENAME (FILE)

Identifies the Master File.

SUFFIX

Identifies the data adapter needed to interpret the request.

The syntax is

```
FILE[NAME]=file, SUFFIX=SQLIDMS [ , $ ]
```

where:

file

Is the filename for the Master File. The filename can consist of a maximum of eight alphanumeric characters. The filename should start with a letter and be representative of the table or view contents.

SQLIDMS

Is the value for the IDMS/SQL data source.

Syntax

How to Declare Segment Attributes

Each table described in a Master File requires a segment declaration. The segment declaration consists of at least two attributes:

SEGNAME

Identifies one table.

SEGTYPE

Identifies the physical storage of rows and the uniqueness of column values.

The syntax for a segment declaration is

```
SEGNAME=segname, SEGTYPE=S0 [ , $ ]
```

where:

segname

Is the segment name that serves as a link to the actual IDMS/SQL table name. It can consist of a maximum of 8 alphanumeric characters. It may be the same as the name chosen for FILENAME, the actual table name, or an arbitrary name.

The SEGNAME value in the Master File must be identical to the SEGNAME value specified in the Access File.

S0

Indicates that IDMS/SQL assumes responsibility for both physical storage of rows and the uniqueness of column values (if a unique index or calc key exists). It always has a value of S0 (S zero).

Syntax **How to Declare Field Attributes**

Each row in a table may consist of one or more columns. These columns are described in the Master File as fields with the following primary field attributes:

FIELDNAME

Identifies the name of a field.

ALIAS

Identifies the full IDMS/SQL column name.

USAGE

Identifies how to display a field on reports.

ACTUAL

Identifies the IDMS/SQL data type and length in bytes for a field.

MISSING

Identifies whether a field supports null data.

Note: You can get values for these attributes from the IDMS/SQL schema definition or standard IDMS/SQL dictionary reports.

The syntax for a field declaration is

```
FIELD[NAME]=fieldname, [ALIAS=]sqlcolumn, [USAGE=]display_format,  
[ACTUAL=]storage_format [, MISSING={ON|OFF}], $
```

where:

fieldname

Is the unqualified name of the field. This value must be unique within the Master File. The name can consist of a maximum of 48 alphanumeric characters (including any filename and segment name qualifiers and qualification characters you may later prefix to them in your requests). The name must begin with a letter. Special characters and embedded blanks are not recommended. The order of field declarations in the Master File is significant with regard to the specification of key columns. For more information, see *Primary Key* on page 11-11.

It is not necessary to describe all the columns of the IDMS/SQL table in your Master File.

sqlcolumn

Is the full IDMS/SQL column name (the data adapter uses it to generate SQL statements). This value must comply with the same naming conventions described for field names.

display_format

Is the display format. The value must include the field type and length and may contain edit options.

The data type of the display format must be identical to that of the ACTUAL format. For example, a field with an alphanumeric USAGE data type must have an alphanumeric ACTUAL data type.

Fields or columns with decimal or floating-point data types must be described with the correct scale (s) and precision (p). Scale is the number of positions to the right of the decimal point. Precision is the total length of the field.

For the server, the total display length of the field or column includes the decimal point and negative sign. In SQL, the total length of the field or column excludes positions for the decimal point and negative sign.

For example, a column defined as DECIMAL(5,2) would have a USAGE attribute of P7.2 to allow for the decimal point and a possible negative sign.

storage_format

Is the storage format of the IDMS/SQL data type and length, in bytes, for the field. For more information on data type support, see the *iWay SQL Reference* manual.

ON

Displays the character specified by the NODATA parameter for missing data. For more information, see *MISSING Attribute* on page 11-9.

OFF

Displays blanks or zeroes for fields having no value. This is the default. For more information, see *MISSING Attribute* on page 11-9.

MISSING Attribute

In a table, a null value represents a missing or unknown value; it is not the same as a blank or a zero. For example, a column specification that allows null values is used where a column need not have a value in every row (such as a raise amount in a table containing payroll data).

Note:

- The default NODATA character is a period (.).
- A column in an IDMS/SQL table that allows null data does not need to include the NULL clause in its table definition, since that is the default. In the Master File for that table, the column that allows null data must be described with the MISSING attribute value ON. The default for this attribute is OFF, which corresponds to the NOT NULL attribute in the IDMS/SQL table definition.
- Null data values appear as zeroes or blanks, if the column allows null data but the corresponding field in the Master File is described with the MISSING attribute value OFF.

Access Files

Each Master File must have a corresponding Access File. The filename of the Access File must be the same as that used for the Master File.

The Access File serves as a link between the server and the data source by providing the means to associate a segment in the Master File with the table it describes. The Access File minimally identifies the table and primary key. It may also indicate the logical sort order of data.

Syntax **How to Establish Segment Declarations**

The segment declaration in the Access File establishes the link between one segment of the Master File and the actual IDMS/SQL table or view. Attributes that constitute the segment declaration are:

SEGNAME

Identifies one table.

TABLENAME

Identifies the table or view. It should contain both the schema and the table name.

DBSPACE

Identifies the IDMS/SQL segment and area in which the IDMS/SQL table is to reside.

KEYS

Identifies how many columns constitute the primary key.

KEYORDER

Identifies the logical sort sequence of data by the primary key.

The syntax for an Access File segment declaration is

```
SEGNAME=segname, TABLENAME=[schema.]tablename, [,DBSPACE=storage,]  
[,KEYS={0|n}] [,KEYORDER={ASC|LOW|HIGH|DESC}] , $
```

where:

segname

Is the 1 to 8 character SEGNAME value from the Master File.

schema

Is the IDMS/SQL schema name for the table or view. It can consist of a maximum of 8 characters. If it is not specified, IDMS/SQL searches for a temporary table definition for the named table. If the named table does not exist, IDMS/SQL uses the current schema in effect for the current user session.

tablename

Is the name of the table or view. It can consist of a maximum of 18 characters.

Note: If any part of the TABLENAME includes a dollar sign (\$), enclose that part in double quotation marks, and enclose the entire TABLENAME value in single quotation marks.

The maximum IDMS/SQL length for a fully-qualified tablename is 36. All names must conform to the rules for identifiers stated in the *CA-IDMS/DB Release SQL Reference* manual.

storage

Is the IDMS/SQL segment and area name (in the form segment.area). If not specified, IDMS/SQL uses the default area associated with the schema. Enclose it in double quotation marks if it begins with a number or special character or contains special characters.

The Access File DBSPACE attribute overrides both the SET command and the installation default.

n

Is the number of columns that constitute the primary key. It can be a value between 0 and 64. The default value is 0. For more information, see *Primary Key* on page 11-11.

LOW (ASC)

Indicates an ascending primary key sort sequence. This value is the default. ASC is a synonym for LOW.

HIGH (DESC)

Indicates a descending primary key sort sequence. DESC is a synonym for HIGH.

Primary Key

A table's primary key consists of the column or combination of columns whose values uniquely identify each row of the table. In the employee table, for example, every employee is assigned a unique employee identification number. Each employee is represented by one and only one row of the table, and is uniquely identified by that identification number.

The order of field declarations in the Master File is significant to the specification of key columns. To define the primary key in a Master File, describe its component fields immediately after the segment declaration. You can specify the remaining fields in any order. In the Access File, the KEYS attribute completes the process of defining the primary key.

To identify the primary key, the data adapter uses the number of columns (n) indicated by the KEYS attribute in the Access File and the first n fields described in the Master File.

Typically, the primary key is supported by the creation of a unique index in the SQL language to prevent the insertion of duplicate key values. The data adapter itself does not require any index in the column(s) comprising the primary key (although a unique index is certainly desirable for both data integrity and performance reasons).

Data Type Support

The following chart provides information about the default mapping of IDMS/SQL data types to server data types:

SQLIDMS Data Type	Data Type		Remarks
	Usage	Actual	
Binary(1..32,760)	A1	A1	$n < 257$
Character(1..32,760)	An	An	n up to 32000
Date	YYMD	DATE	
Decimal (1..31,0..31)(n,m)	P(n+2,m)	P8	$n \leq 14$
Decimal (1..31,0..31)(n,m)	P(n+2,m)	P16	$n > 14$
Double Precision	D20.2	D8	
Float (1..56)(n)	F9.2	F4	$n \leq 24$
Float (1..56)(n)	D20.2	D8	$n > 24$
Graphic (1..16380)(n)	A(2*n)	Kn	
Integer	I11	I4	
Longint/BIGINT	I11	I4	
Numeric (1..31,0..31)(n,m)			Same as decimal
Real	F9.2	F4	
Smallint	I6	I4	
Time	A8	A8	
Timestamp	A26	A26	
Varchar (1..32760)(n)	An	An	$n \leq 32000$
Vargraphic (1..16379)(n)	TX	TX	

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to a server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Syntax

How to Override Default Precision and Scale

```
ENGINE [SQLIDMS] SET CONVERSION RESET
ENGINE [SQLIDMS] SET CONVERSION format RESET
ENGINE [SQLIDMS] SET CONVERSION format [PRECISION nn [mm]]
ENGINE [SQLIDMS] SET CONVERSION format [PRECISION MAX]
```

where:

SQLIDMS

Indicates the SQLIDMS data source. You can omit this value if you previously issued the SET SQLENGINE command.

RESET

Returns any previously specified precision and scale values to the data adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

format

Is any valid format supported by the data source. Possible values are:

INTEGER which indicates that the command applies only to INTEGER.

DECIMAL which indicates that the command applies only to DECIMAL columns.

REAL which indicates that the command applies only to single precision floating point columns.

FLOAT which indicates that the command applies only to double precision floating point columns.

nn

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (see description of MAX).

mm

Is the scale. This is valid with DECIMAL, REAL, and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. If the scale is not required, you must set mm to zero.

MAX

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
REAL	9
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example **Setting the Precision and Scale Attributes**

The following example shows how to set the precision attribute for all INTEGER fields to 7:

```
ENGINE SQLIDMS SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLIDMS SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER fields to the default:

```
ENGINE SQLIDMS SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLIDMS SET CONVERSION RESET
```


Customizing the IDMS/SQL Environment

The Data Adapter for IDMS/SQL provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Setting the User-specific Schema Name

The Data Adapter for IDMS/SQL uses the user-specified schema name as the first qualifier for all SQL requests involving SQL tables or views. This command overrides the IDMS/SQL current schema in effect and precludes the specification of unqualified table names. This prevents passing of unqualified table names to IDMS/SQL.

Syntax **How to Set SESSION**

```
ENGINE [SQLIDMS] SET SESSION CURRENT SCHEMA schema
```

where:

SQLIDMS

Indicates the IDMS/SQL data source. You can omit this value if you previously issued the SET SQLENGINE command.

schema

Is the name of the schema in SQL requests.

Controlling Transactions

IDMS/SQL protects data being read by one user from changes (INSERT, UPDATE, or DELETE) made by others; the Isolation Level setting governs the duration of the protection. That is, the Isolation Level determines when shared locks on rows are released, so that those rows or pages become available for updates by other users. IDMS/SQL allows you to dynamically set the Isolation Level within the server session using the IDMS/SQL SET TRANSACTION command.

The SET TRANSACTION CURSOR STABILITY or TRANSIENT READ command affects the duration of row or page shared locks on IDMS/SQL tables for the duration of the IDMS/SQL transaction. You can specify the command within a stored procedure. The setting remains in effect for the server session or until you reset it.

Syntax **How to Control Transactions**

```
ENGINE [SQLIDMS] SET TRANSACTION {CURSOR_STABILITY|TRANSIENT READ|READ ONLY|READ WRITE}
```

where:

SQLIDMS

Indicates the IDMS/SQL data source. You can omit this value if you previously issued the SET SQLENGINE command.

CURSOR_STABILITY

Provides the maximum amount of concurrency while guaranteeing the integrity of the data selected. This is the default.

TRANSIENT READ

Allows the reading of records locked by other users. This is recommended for SQL request only. Transient read prevents the SQL transaction from performing updates. Use this only when you do not need the data retrieved to be absolutely consistent and accurate. If you specify Transient Read, IDMS/SQL assumes it is read-only.

READ ONLY

Allows data to be retrieved, but does not allow the data source to be updated.

READ WRITE

Allows data to be retrieved, and allows the data source to be updated.

Designating a Default Tablespace

You can use the SET DBSPACE command to designate a default tablespace for tables you create.

For the duration of the session, the data adapter places these tables in the IDMS/SQL tablespace that you identify with the SET DBSPACE command. If the SET DBSPACE command is not used, IDMS/SQL uses the default tablespace for the connected user.

Syntax How to Set DBSPACE

```
ENGINE [SQLIDMS] SET DBSPACE storage
```

where:

SQLIDMS

Indicates the IDMS/SQL data source. You can omit this value if you previously issued the SET SQLENGINE command.

storage

Is the segment.area name that will contain the IDMS/SQL tables created by the CREATE FILE command. If a name is not specified, the table will be placed in the IDMS/SQL default area for the current schema in effect for the user's SQL session.

Note: This command will only affect CREATE TABLE requests made by Table Services. It does not affect Passthru CREATE TABLE commands.

Overriding Default Parameters for Index Space

You can use the SET IXSPACE command to override the default parameters for the IDMS/SQL index space implicitly created by the CREATE FILE and HOLD FORMAT SQLORA commands.

Syntax How to Set IXSPACE

```
ENGINE [SQLIDMS] SET IXSPACE [index-spec]
```

where:

SQLIDMS

Indicates the IDMS/SQL data source. You can omit this value if you previously issued the SET SQLENGINE command.

index-spec

Is the portion (up to 94 bytes) of the SQL CREATE INDEX statement beginning with IN segment.area (as specified in the *CA-IDMS/DB Reference* CREATE INDEX syntax diagram).

Note: To reset to the IDMS/SQL default index space parameters, issue the SET IXSPACE command with no operands.

Example Specifying Segment.Area Name and Index Block

The following example shows how to set the segment.area name and INDEX BLOCK of the CREATE INDEX statement with IXSPACE:

```
ENGINE SQLIDMS  
SET IXSPACE IN EMPSEG.EMPAREA INDEX BLOCK CONTAINS 5 KEYS  
END
```

You can use the SQL ? query command to determine the current IXSPACE setting. If the current setting is the default, IXSPACE does not display in the SQL SQLIDMS ? output.

Note: This command will only affect CREATE INDEX requests made by Table Services. It does not affect Passthru CREATE INDEX commands.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru UPDATE or DELETE command.

Syntax How to Obtain the Number of Rows Updated or Deleted

```
ENGINE [SQLIDMS] SET PASSRECS {ON|OFF}
```

where:

SQLIDMS

Indicates the IDMS/SQL data source. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides an informational message after the successful execution of an SQL Passthru UPDATE or DELETE command. This value is the default.

OFF

Does not provide a message after the successful execution of an SQL Passthru UPDATE or DELETE command.

In addition, the data adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Note that since, by definition, the successful execution of an INSERT command always affects one record, INSERT does not generate this information.

CHAPTER 12

Getting Started in IMS

Topics:

- Preparing the IMS Environment
- Configuring the Data Adapter for IMS
- Managing IMS Metadata
- Migrating From an Existing MVS Server

The Data Adapter for IMS allows applications to access IMS data sources. The adapter converts application requests into native IMS statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

The Data Adapter for IMS is supported on the OS/390 and z/OS platform and uses the IMS DBCTL environment to connect to IMS databases.

Preparing the IMS Environment

Before configuring the Data Adapter for IMS, verify that your site is running IMS Version 3.1 or higher and that all APPLCTN macros describing PSBs that will be accessed by multiple users specify SCHDTYP=Parallel. If necessary, modify the APPLCTN macros for such PSBs to include the attribute SCHDTYP=Parallel, and rerun the IMS SYSGEN to make the changes effective.

The Data Adapter for IMS connects using the DBCTL environment. For the connection to be made, the following libraries must be identified and allocated to the STEPLIB in the ISTART JCL member of *qualif.release.servertype.DATA*:

- The DFHPZP library.
- The SDFSRESL library.

Note: *qualif* is provided by the user. The *release* and *servertype* vary with the release and license key being used. Refer to the Server Installation for OS/390 and z/OS in the *iWay Server Installation* manual for further information on the configuration dataset naming convention.

The PZP library must have a member that has been generated to identify the correct DBCTL environment for the connection. The following steps must be completed before the server can be configured, using the Web Console, for the Data Adapter for IMS:

1. Create the DRA startup table.
2. Assemble and link the DRA startup table.
3. Verify that the DBCTL is operational.
4. Establish security.

Syntax

How to Create the DRA Startup Table: DFSPZPxx

The Database Resource Adapter (DRA) is the interface between a user task and DBCTL. The DRA Startup Table contains values that define the characteristics of the DRA. The DRA Startup Table is named

`DFSPZPxx`

where:

`xx`

Is a two-character suffix that should be chosen to comply with your site's standards.

Once this suffix has been chosen, its value is needed during the configuration phase using the Web Console.

Example Assembling and Linking the DRA Startup Table

This sample JCL illustrates how to assemble and link the DFHPZPxx member. You can normally find sample JCL in your IMS installation library. For a list of required keywords and the descriptions, see *Keywords* on page 12-4.

```
//job card goes here
//ASSEMBLE EXEC PGM=IEV90,REGION=2M,PARM='OBJECT,NODECK'
//SYSLIB DD DSN=IMS.MACLIB,DISP=SHR
//SYSLIN DD UNIT=SYSDA,DISP=(,PASS),
//          SPACE=(80,(100,100),RLSE),
//          DCB=(BLKSIZE=80,RECFM=F,LRECL=80)
//SYSPRINT DD SYSOUT=*,DCB=BLKSIZE=1089
//SYSUT1 DD UNIT=SYSDA,DISP=(,DELETE),
//          SPACE=(CYL,(10,5))
//SYSIN DD *
PRP TITLE 'DATABASE RESOURCE ADAPTER STARTUP PARAMETER TABLE'
DFSPZPxx CSECT
        EJECT
        DFSPRP DSECT=NO, X
                DBCTLID=IMS3, X
                DDNAME=DFSRESLB, X
                DSNAME=IMS.RESLIB, X
                CNBA=150, X
                MAXTHRD=150, X
                MINTHRD=5 X
        END
/*
/**
//LINK EXEC PGM=IEWL,PARM='XREF,LIST',COND=(0,LT,ASSEMBLE),
//          REGION=4M
//SYSLIN DD DSN=*.ASSEMBLE.SYSLIN,DISP=(OLD,DELETE)
//SYSPRINT DD SYSOUT=*,DCB=BLKSIZE=1089
//SYSUT1 DD UNIT=(SYSDA,SEP=(SYSLMOD,SYSLIN)),
//          SPACE=(1024,(100,10),RLSE),DISP=(,DELETE)
//SYSLMOD DD DISP=SHR,DSN=qualif.release.HOME.LOAD(DFSPZPxx)
```

where:

qualif

Is the high level qualifier for the data sets.

xx

Is the two character suffix chosen for the DRA Startup Table.

Note: The *qualif.release.HOME.LOAD* library must be concatenated ahead of *IMS.RESLIB* in the allocation for DDNAME STEPLIB in *qualif.release.servertype.DATA(ISTART)*.

Reference Keywords

The following chart describes the keywords required to generate the DFHPZP library. Other keywords that affect your IMS environment, and may be required at your site, are described in the *IBM IMS System Definition Reference*.

Keyword	Description	Default
AGN	Is a 1 to 8 character application group name used as part of the DBCTL security function. For more information on DBCTL security, see the <i>IMS System Administration Guide</i> .	N/A
CNBA	Is the total number of Fast Path buffers for the server's use. For a description of Fast Path DEDB buffer usage, see the <i>IMS System Administration Guide</i> . You can omit this parameter if your site does not utilize Fast Path.	N/A
DBCTLID	Is the 4-character name of the DBCTL region. This is the same as the IMSID parameter in the DBC procedure. For more information on the DBC procedure, see the <i>IBM IMS System Definition Reference</i> .	SYS1
DDNAME	Must be DFSRESLB. It is the DDNAME that will be allocated to the DBCTL RESLIB library (see the DSNAME keyword).	N/A
DSNAME	Is the 1- to 44-character data set name of the DBCTL RESLIB library. It must contain the DRA modules and be MVS authorized.	IMS.RES LIB
FPBOF	Is the number of Fast Path DEDB overflow buffers to be allocated per thread. For a description of Fast Path DEDB buffer usage, see the <i>IMS System Administration Guide</i> . You can omit this keyword if your site does not use Fast Path.	00
FPBUF	Is the number of Fast Path DEDB buffers to be allocated and fixed per thread. For a description of Fast Path DEDB buffer usage, see the <i>IMS System Administration Guide</i> . You can omit this parameter if your site does not utilize Fast Path.	00
FUNCLV	Is the level of the DRA that the CCTL supports. FUNCLV=1 means the CCTL uses the DRA at the IMS 3.1 level.	1
MAXTHRD	Is the maximum number of concurrent DRA threads. The value cannot exceed 255.	1

Keyword	Description	Default
MINTHRD	Is the minimum number of concurrent DRA threads available. Since the number of threads specified by this keyword will be allocated at all times, regardless of whether they are used, be careful when selecting this value. The value cannot exceed 255.	1
SOD	Is the output class to use for a SNAP DUMP of abnormal thread termination.	A
TIMEOUT	Is the number of seconds a CCTL should wait for the successful completion of a DRA TERM request. Specify this value only if the CCTL is coded to use it. This value is returned to the CCTL upon completion of an INIT request.	60
TIMER	Is the number of seconds between attempts of the DRA to identify itself to DBCTL during an INIT request.	60
USERID	Is the 8-character name of the CCTL region. No two CCTLs (servers accessing the same IMS region through DBCTL) can have the same USERID (address space ID).	N/A

Syntax **How to Verify the DBCTL Is Operational**

The following JCL can be used to verify that the DBCTL is properly installed. Run this job after making any changes necessary to the JCL to suit your site's standards and naming conventions.

```

/*****
/* Substitutions:
/*
/*      qualific  Change to the high level qualifier
/*              for your data sets.
/*
/*      edapsb1   Change to any PSB name.
/*
/*      pcicspsb  RACF class to be used in the test. If this
/*              parameter is omitted, security checking will
/*              not be performed.
/*
/*      xx        Is the 2 character suffix that identifies the DRA
/*              startup module.
/*
/*****
//STEP1      EXEC   PGM=IMDTEST,PARM='emppsbl,pcicspsb,xx' ,
//              REGION=4096K,TIME=(1,5)
//STEPLIB    DD     DSN=qualific.release.HOME.LOAD,DISP=SHR
//              DD     DSN=IMS.RESLIB,DISP=SHR
//SYSIN      DD     DUMMY
//SYSOUT     DD     SYSOUT=*
//SYSPRINT   DD     SYSOUT=*

```

where:

emppsbl

Is a PSB name.

pcicspsb

Is a security class.

xx

Is a 2-character suffix that identifies the DRA Startup Module to be loaded as described in *How to Create the DRA Startup Table: DFSPZPxx* on page 12-2.

Note: If you omit this parameter, its value defaults to 00, which may cause IMS to load the incorrect module.

qualific

Is the high level qualifier for the data sets.

After running the JCL, examine the JES output. The return codes displayed on the following lines indicate that DBCTL is properly installed:

```

12.45.47 JOB05736 +CCTL: JOBNAME(USERID1) PSB(EMPPSB1) CLASS(PCICSPSB)
12.45.47 JOB05736 +CCTL: INIT CALLED
12.45.47 JOB05736 +CCTL: INIT RETURNED,RETC(00000000)
12.45.47 JOB05736 +CCTL: WAITING FOR CONTROL EXIT
12.45.47 JOB05736 +CTLX: CONTROL EXIT CALLED
12.45.47 JOB05736 +CTLX: CONTROL EXIT RETURNED
12.45.47 JOB05736 +CCTL: INIT COMPLETED, FUNC(02) SFNC(00) RCOD(00)
RETC(00000000)
12.45.47 JOB05736 +CCTL: SECURITY CHECK COMPLETED, RC(00000004)
12.45.47 JOB05736 +CCTL: TERMINATE DRA CALLED
12.45.47 JOB05736 +CTLS: SUSPEND EXIT CALLED
12.45.47 JOB05736 +CTLR: RESUME EXIT CALLED
12.45.47 JOB05736 +CTLS: SUSPEND EXIT RETURNED
12.45.47 JOB05736 +CTLR: RESUME EXIT RETURNED
12.45.47 JOB05736 +CCTL: TERMINATE DRA COMPLETED, RETC(00000000)

```

Note that the installation is successful even when the security check completes with a return code of 4.

If the JES output indicates an error, make sure that a DRA Startup Table with a suffix of 00 exists. See *How to Create the DRA Startup Table: DFSPZPxx* on page 12-2. If it exists, verify that:

- The parameters specified in it (especially DSNNAME and DDNAME) are correct.
- The DBCTLID keyword points to the correct IMS system.
- DBCTL is installed at your site, and that it is functioning correctly.

Establishing Security

The DBCTL environment, when accessed through the server, enables the use of security systems through the standard SAF interface. With the SAF interface, your site can use security products such as RACF, CA TOP SECRET, and CA ACF2 to restrict access to PSBs. Before allowing access to a particular PSB, the security system verifies that the user is authorized to read the PSB.

Note: The DBCTL function is tested and verified with the RACF product. Other SAF products using identical calls should perform properly when installed and verified by your site's security administrator.

RACF comes with several predefined security classes. Customer sites can use an existing class (such as PCICSPSB) or define a resource class specifically for DBCTL use.

Syntax **How to Define a PSB Resource**

The following syntax illustrates how to define a PSB resource through a PCICSPSB profile to RACF, and how to grant users permission to access the resource.

```
RDEFINE PCICSPSB (psbname) UACC(NONE) NOTIFY(sys_admin_userid)  
PERMIT psbname CLASS(PCICSPSB) ID(user_or_group [user_or_group ...])  
ACCESS(READ)
```

where:

psbname

Is a PSB name to be protected by RACF.

sys_admin_userid

Is the user ID of the system administrator.

user_or_group

Authorizes the user IDs and/or user groups listed in the PERMIT command to read the specified PSB. Separate items in the list with blanks.

At run time, after the PSB is selected, but prior to scheduling it, the server issues a call to the security system and verifies that the user is authorized to read the PSB.

Configuring the Data Adapter for IMS

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish. The Data Adapter for IMS is configured from the Web Console.

Procedure How to Configure the Data Adapter From the Web Console

1. Start the Web Console and select *Adapters* from the left pane.
2. Expand the Add folder.
3. Expand the IMS folder and click *IMS*. The configuration window displays the following fields:

Field	Description
IMS Sec	ON activates DBCTL security. This setting requires that the server be in authorized mode. OFF does not implement DBCTL security.
IMS DBCTL	START.
IMS Class	Points to a valid class that contains the security rule. The Systems Administrator can define any class name required by the security implementation of the site. The default value is PCICSPSB.
IMS Pzp	The suffix of the DFSPZP module created during the generation of the Data Adapter for IMS/DBCTL.

4. Supply valid values and click *Configure*.
5. Restart the server so the connection to DBCTL can be initialized.

If you do not have existing synonyms (Master Files), proceed to *Managing IMS Metadata* on page 12-10. If existing synonyms are available through JCL allocations, the Data Adapter for IMS is ready for use.

Dynamic Setting of PSB

The PSB name to use for access can be set dynamically with the following command

```
ENGINE IMS SET PSB psbname
```

where:

psbname

Is the PSB name you wish to dynamically set.

Managing IMS Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the IMS data types.

For the Data Adapter for IMS to access IMS files, you must describe each IMS file you use in a Master File and Access File. The logical description of an IMS file is stored in a Master File, which describes the field layout.

Creating Synonyms

Synonyms define unique names (or aliases) for each IMS file that is accessible from a server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

Procedure How to Create a Synonym Through the Web Console

1. Start the Web Console and click *Metadata* from the left pane.
2. Expand the Add folder and click *IMS*.

The following window opens:

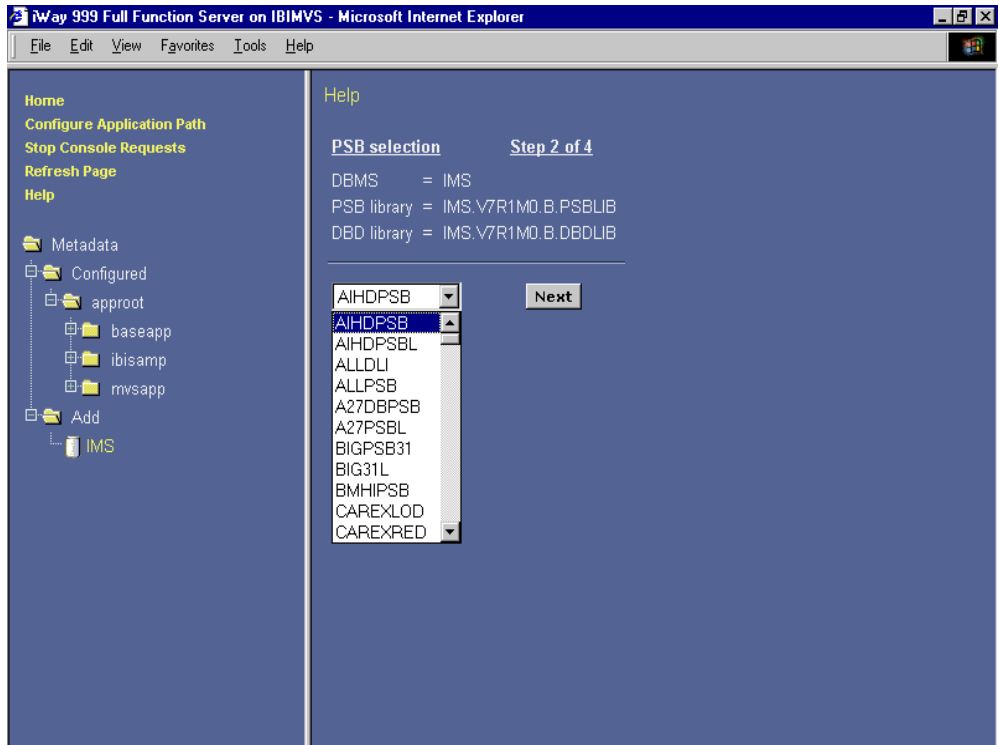
The screenshot shows a web browser window with the address bar displaying `http://bimvsx:5056/metadata.html`. The page content is as follows:

- Left sidebar:
 - Home
 - Configure Application Path
 - Stop Console Requests
 - Refresh Page
 - Help
 - Metadata
 - Configured
 - approot
 - Add
 - IMS

- Main content area:
- Header: Please provide fully qualified MVS library names for: Step 1 of 4
- Form fields:
 - PSB library name: (Sample: IMS.V7R1M0.B.PSBLIB)
 - DBD library name: (Sample: IMS.V7R1M0.B.DBDLIB)
- Next button

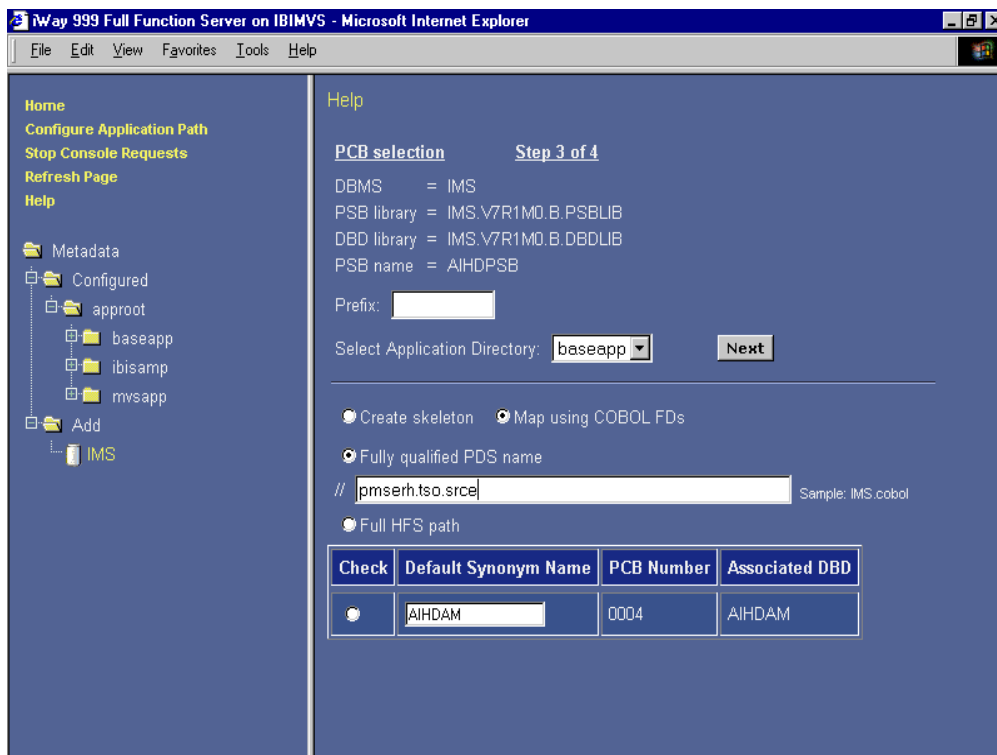
3. Enter the MVS library names that correspond to the IMS PSBLIB and DBDLIB in the respective fields. These must be fully qualified MVS dataset names.
4. Click *Next*.

The following window opens:



5. From the drop-down box select the PSB member name.
6. Click Next.

The following window opens:

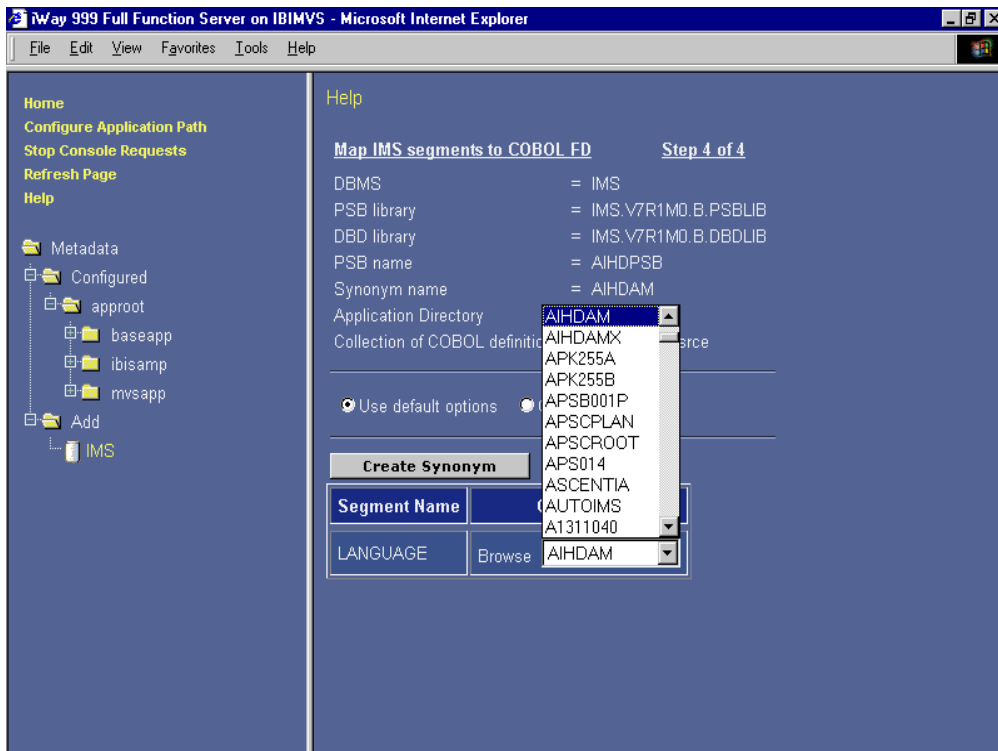


7. From the *Select Application Directory* drop down box, select the directory where the resultant Master File and Access File will be saved.
8. Select the *Map using COBOL FDs* option button if you have a Cobol FD library available that describes the PCB view. Enter the HFS path or fully qualified MVS Library (preceded by //) in the box provided. The *with selection* box can be used to filter the selection for members from the PDS or file suffix from the HFS directory. If no selection value is entered, all members/files will be selected.

Note: If you select the *Create skeleton* option, the final screen opens without the option to select a Cobol FD entry. In this case, click *Create Synonym* to complete the process.

9. Select the PCB you want to use by clicking the option button in the check column.
10. Click *Next*.

- From the final window (shown below) you can associate or map the IMS segments to the corresponding COBOL FDs.



- From the drop-down list under the COBOL FD column, select the COBOL FD that will be used for each segment name.
- Click *Create Synonym*. The result of the Create Synonym process will be a Master File and an Access File. If you want to customize how the COBOL FD is translated, click *Customize options* before you click *Create Synonym*. Each option has help available.

Syntax How to Create a Synonym Manually

```
CREATE SYNONYM appname/synonym FOR psbname DATABASE 1,2,3,etc. DBMS IMS
AT psbllib/dbdlib PARMS [" < string 'fdparms' > "]
END
```

where:

appname

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, this application name must not be used.

FOR

Indicates the *psbname* that contains a PCB reference to the database being processed.

DATABASE

Refers to the numeric position of the database PCB within the PSB.

AT

Indicates the corresponding MVS dataset names for the PSBLIB and the DBDLIB. Both must be specified. The dataset names must be fully qualified MVS loadlib names and are delimited with a slash.

PARMS

Contains the location of the individual corresponding COBOL FD (within double quotes); the associated database segment; and parameters specific to COBOL FDs (within single quotes).

The location may be an HFS file—/u/pgmjks/fds/patinfo/PATINFO, or an MVS PDS (partitioned dataset)—'PMSERH.TSO.SRCE(PATINFO)'. The syntax is

```
<fd name><comma><segment name><space><fd name><comma><segment name>
etc.
```

fdparms

Indicates COBOL FD specific parameters, such as:

```
[SKIPHPHEN = {<number> | all}]
      [UNDERBARS]
      [ORDER = {Y | N}]
      [REDEFINE = {comment | segm}]
      [LEVEL88 = {comment | skip}]
      [OCCURS = {segm | field}]
      [EDITFIELDS = {S,C,B,R,M,N,L}]
      [ZONEFIELDS = {packed | alpha}]
```

If no FDPARMS are entered, the underlined defaults will be in effect.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

CREATE SYNONYM creates a Master File and Access File, which represent the server metadata.

Example Using CREATE SYNONYM

Use the following syntax to create a synonym for the variable "AIHDAM":

```
CREATE SYNONYM AIHDAM FOR AIHDPSB DATABASE 4 DBMS IMS AT
IMS.V7R1M0.B.PSBLIB/IMS.V7R1M0.B.DBDLIB
    PARMS "'/'PMSERH.TSO.SRCE(AIHDAM)',AIHDAM
    FDPARMS = 'SKIPHYPHEN 0 UNDERBARS ONERROR C ORDER N REDEFINE S
LEVEL88 N OCCURS Y ZONEFIELDS A'"
END
CREATE SYNONYM AIHDAM FOR AIHDPSB DATABASE 4 DBMS IMS AT
IMS.V7R1M0.B.PSBLIB/IMS.V7R1M0.B.DBDLIB
    PARMS "/u/pgmjisk/fds/aihdam,AIHDAM
    FDPARMS = 'SKIPHYPHEN 0 UNDERBARS ONERROR C ORDER N REDEFINE S
LEVEL88 N OCCURS Y ZONEFIELDS A'"
END
```

Generated Master File aihdam.mas

```
FILENAME=AIHDAM, SUFFIX=IMS , $
SEGMENT=LANGUAGE, SEGTYPE=S0, $
$ GROUP=AIHDAM_IO, USAGE=A19, ACTUAL=A19, $
FIELDNAME=EMPL6, ALIAS=EMPL6.HKY, USAGE=A4, ACTUAL=A4, $
FIELDNAME=LANG6, ALIAS=LANG6.IMS, USAGE=A15, ACTUAL=A15, $
GROUP=IXEMP6.SKY, ALIAS=IXEMP6.SKY, USAGE=A4, ACTUAL=A4, $
FIELDNAME=IX_EMPL6 , ALIAS=EMPL6, USAGE=A4, ACTUAL=A4, $
```

Generated Access File aihdam.acx

```
PSB=AIHDPSB,PCBNUMBER= 4,PL1=NO,WRITE=NO,$
SEGNAME=LANGUAGE, KEYTYPE=S1 , $
ALTPCBNAME=IXEMP6 , ALTPCBNUMBER=5,$
```

Migrating From an Existing MVS Server

Migrating from an existing MVS server environment to OS/390 and z/OS consists of the following steps:

1. Identify the source DSNs for your existing IMS server's Master File and Access File.
2. Identify the DSN for your existing IMS server's FOCPSB file.
3. Update the *qualif.release.servertype.DATA(ISTART)* JCL with the identified DSNs allocated to MASTER, ACCESS, and FOCPSB, respectively.
4. Duplicate any IMS SET commands from the existing server profile into the new *edasprof.prf* file using the Web Console *Workspace/Edit Files* option.
5. Restart the server.

If you have not already configured the server for DBCTL, configure it now; otherwise the server is ready for use.

CHAPTER 13

Getting Started in Informix

Topics:

- Preparing the Informix Environment
- Configuring the Data Adapter for Informix
- Managing Informix Metadata
- Customizing the Informix Environment
- Calling an Informix Stored Procedure Using SQL Passthru

The Data Adapter for Informix allows applications to access Informix data sources. The adapter converts application requests into native Informix statements and returns optimized answer sets to the requesting application. This adapter has read/write capabilities, which enables the adapter to insert the data from an application to the data source.

Preparing the Informix Environment

The Informix Data Adapter minimally requires the installation of the Informix Client SDK software. The Informix Client SDK software allows you to connect to a local or remote Informix database server.

In order to support a greater range of Informix DBMS releases, the current server release uses Informix Client SDK in its Data Adapter for Informix. All the current Informix Client SDK's support connectivity to wide range of Informix databases. For example, SDK 2.6 will support connectivity to Informix releases 5.1 through 9.3.

Note: Always check the Informix SDK documentation for supported releases of Informix databases.

The following environment settings apply when configuring the server with the Data Adapter for Informix.

Procedure How to Set Up the Environment on Windows NT/2000

On Windows NT/2000, the Informix environment is set up during the installation of Informix Server and Informix Client SDK.

Procedure How to Set Up the Environment on UNIX

1. Identify the Informix Client SDK using the UNIX environment variable `$INFORMIXDIR`. For example, to set the home directory for the Informix software to `/usr/informix930`, specify:

```
INFORMIXDIR=/usr/informix930
export INFORMIXDIR
```

2. Identify the Default Informix Server using the UNIX environment variable `$INFORMIXSERVER`:

```
INFORMIXSERVER=server_name
export INFORMIXSERVER
```

3. Specify the path to the Informix shared library using the UNIX environment variable `$LD_LIBRARY_PATH`. For example:

```
LD_LIBRARY_PATH=$INFORMIXDIR/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH
```

4. Specify the format of the DATE fields using the UNIX environment variable `$DBDATE`. For example, if the user wants the DATE to be returned, in the following format 01/08/2001:

```
DBDATE=MDY4/
export DBDATE
```


Accessing a Remote Informix Server

Using the standard rules for deploying the Informix Client, the server supports connections to:

- Local Informix database servers.
- Remote Informix database servers. To connect to a remote Informix database server, the `sqlhosts` file on the source machine must contain an entry pointing to the target machine and the listening process must be running on the target machine.

Informix SQL ID (for Informix SE only)

When you access Informix SE, you are identified by your UNIX login ID. The UNIX ID becomes the owner ID for any Informix objects (such as tables or indexes) created with immediate SQL commands. Immediate commands are non-parameterized SQL statements that are passed directly to the data source. The UNIX ID is also used as the sole authorization ID for Informix GRANT and REVOKE statements.

Other types of requests, such as SQL SELECTs, sponsor an Informix search for the necessary privileges to act on the particular tables and views in a data source using the UNIX ID. All Informix security rules are respected.

The following table shows the UNIX privileges required in order to specify certain SQL statements.

Statement	Requires
SELECT	Read permission for the Informix data and index files. UNIX write and execute (search) privileges for the data source directory.
UPDATE	Write permission for the Informix data and index files. UNIX write and execute (search) privileges for the data source directory.

ANSI-Compliant Data Sources

If you create a data source to be ANSI-compliant, owner naming is enforced by Informix. Therefore, to preserve any lowercase owner name, you must enclose it in double quotation marks. Otherwise, Informix will default to uppercase for the owner name. For example:

```
ENGINE SQLINF
SELECT * FROM "user1".test
END
```

For additional information, see the *Informix Guide to SQL*.

XA Support

Read/write applications accessing Informix data sources are able to perform transactions managed in XA-compliant mode.

To activate the XA Transaction Management feature, the server has to be configured in Transaction Coordination Mode, using the Web console configuration functions. Using Transaction Coordination Mode guarantees the integrity of data modification on all of the involved DBMSs and protects part of the data modifications from being committed on one DBMS and terminated on another.

For complete documentation on XA compliance, see the XA Support appendix.

Configuring the Data Adapter for Informix

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to an Informix database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Manually add the command in the global server profile (edasprof.prf) or in a user profile (user.prf).
- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the global server profile. For more information, see *Declaring Informix Connection Attributes from the Web Console*.

You can declare connections to more than one Informix database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to Informix Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Syntax **How to Declare Connection Attributes Manually**

Explicit authentication. The user ID and password are explicitly specified for each connection and passed to Informix, at connection time, for authentication. The syntax is

```
ENGINE [SQLINF] SET CONNECTION_ATTRIBUTES [connection]
[data_source]/user_ID,password [;dbname]
```

Password passthru authentication. The user ID and password are explicitly specified for each connection and passed to Informix, at connection time, for authentication. This option requires that the server be started with security off. The syntax is

```
ENGINE [SQLINF] SET CONNECTION_ATTRIBUTES [connection]
[data_source]/[;dbname]
```

Trusted authentication. The adapter connects to Informix as a Windows login using the credentials of the Windows user impersonated by the server data access agent. The syntax is

```
ENGINE [SQLINF] SET CONNECTION_ATTRIBUTES [connection]
[data_source]/, [;dbname]
```

where:

SQLINF

Indicates the Data Adapter for Informix. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is a logical name (or a data source name) used to identify this particular set of attributes.

If you plan to have only one connection to Informix, this parameter is optional. If not specified, the local database server serves as the default connection.

data source

Is the name of the database server as defined by the dbservername field in the sqlhosts file.

If data_source is not specified, the Informix default server (value of the INFORMIXSERVER environment variable) is used. You can also specify the default server with two consecutive single quotation marks.

user_ID

Is the primary authorization ID by which you are known to Informix.

password

Is the password associated with the primary authorization ID.

dbname

Also referred to as *schema*, is the name of the Informix database used for this connection. The database name, including path, must be included in single quotes.

Example Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command allow the application to access the Informix database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLINF SET CONNECTION_ATTRIBUTES SAMPLESERVER/MYUSER,PASS
```

The following SET CONNECTION_ATTRIBUTES command connects to the Informix database server named SAMPLESERVER using Password Passthru authentication:

```
ENGINE SQLINF SET CONNECTION_ATTRIBUTES SAMPLESERVER/
```

The following SET CONNECTION_ATTRIBUTES command connects to a local Informix database server using operating system authentication:

```
ENGINE SQLINF SET CONNECTION_ATTRIBUTES /,
```

Reference Declaring Informix Connection Attributes From the Web Console

Attribute	Description
Connection name	Is a logical name used to identify this particular set of connection attributes.
Database name	Is the Informix database name. For Informix SE, the entire path to the location of the database files including the database name must be specified.
Datasource	Is the Informix database server as defined by the dbservername field in the sqlhosts file.
Security	<p>There are three methods by which a user can be authenticated when connecting to Informix:</p> <p>Explicit. The user ID and password are explicitly specified for each connection and passed to Informix, at connection time, for authentication.</p> <p>Password passthru. The user ID and password received from the application are passed to Informix, at connection time, for authentication. This option requires that the server be started with security off.</p> <p>Trusted. The adapter connects to Informix as a Windows login using the credentials of the Windows user impersonated by the server data access agent.</p>

Attribute	Description
User	Is the name by which the user is known to Informix.
Password	Is the password associated with the user name.

Overriding the Default Connection

Once connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax How to Change the Default Connection

```
ENGINE [SQLINF] SET DEFAULT_CONNECTION [connection]
```

where:

SQLINF

Indicates the Data Adapter for Informix. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the service name defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following error message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following error message is issued: FOC1671, Command out of sequence.

Example Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the Informix database server named SAMPLENAME as the default Informix database server:

```
ENGINE SQLINF SET DEFAULT_CONNECTION SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax **How to Control the Connection Scope**

```
ENGINE [SQLINF] SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

SQLINF

Indicates the Data Adapter for Informix. You can omit this value if you previously issued the SET SQLENGINE command.

FIN

Disconnects automatically only after the session has been terminated. This value is the default.

COMMIT

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Informix Metadata

When the server accesses a data source, it needs to know how to interpret the data that it finds. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Informix data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Informix table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Editing Metadata page opens.

To create a synonym, you must have configured the adapter. See *Configuring the Data Adapter for Informix* for more information.

2. Expand the *Add* folder, expand the adapter folder, and then click a connection. The right pane displays table and view selection options:

Select All Tables/Views. Select this radio button to create synonyms for all tables and views. This value is the default.

Filter by Name, Owner and Table Type. Select this radio button to filter the tables or views for which to create synonyms.

Selecting this option adds the following:

The screenshot shows a dark blue panel with the following elements:

- Two radio buttons: Select all Tables/Views and Filter by Name, Owner and Table Type.
- An 'Owner' label followed by an empty text input field and the text '- Sample: abc%'.
- A 'Table name' label followed by a text input field containing 'NF%' and the text '- Sample: abc%'.
- A 'Table type' label followed by three radio buttons: TABLE, VIEW, and TABLE/VIEW.
- A grey button labeled 'Select Tables' at the bottom left.

Owner. Enter a string for filtering the owners' IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owners' IDs begin with the letters ABC; %ABC to select tables or views whose owners' IDs end with the letters ABC; %ABC% to select tables or views whose owners' IDs contain the letters ABC at the beginning, middle, or end.

Table Name. Enter a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

Table Type. Select one of the following options: Tables (this is the default), Views, Tables/Views.

3. Click *Select Tables*. All tables that meet the specified criteria are displayed:

Prefix: SET CONVERSION LONGCHAR: TEXT (for Web Focus) ▼

Select Application Directory: baseapp ▼

All Select

Check	Default Synonym Name	Owner/Schema	Table Name	Type
<input type="checkbox"/>	<input type="text" value="NF29001"/>	EDAQA	NF29001	TABLE
<input type="checkbox"/>	<input type="text" value="NF29002"/>	EDAQA	NF29002	TABLE
<input type="checkbox"/>	<input type="text" value="NF29003"/>	EDAQA	NF29003	TABLE
<input type="checkbox"/>	<input type="text" value="NF29004"/>	EDAQA	NF29004	TABLE
<input type="checkbox"/>	<input type="text" value="NF29005"/>	EDAQA	NF29005	TABLE
<input type="checkbox"/>	<input type="text" value="NF29006"/>	EDAQA	NF29006	TABLE
<input type="checkbox"/>	<input type="text" value="NF29007"/>	EDAQA	NF29007	TABLE
<input type="checkbox"/>	<input type="text" value="NF29008"/>	EDAQA	NF29008	TABLE
<input type="checkbox"/>	<input type="text" value="NF29009"/>	EDAQA	NF29009	TABLE
<input type="checkbox"/>	<input type="text" value="NF29012"/>	EDAQA	NF29012	TABLE
<input type="checkbox"/>	<input type="text" value="NF29013"/>	EDAQA	NF29013	TABLE

4. If you have tables with identical table names, assign a prefix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave field blank.

5. From the SET CONVERSION LONGCHAR drop-down list, select: Text, Alpha, or BLOB. The default value is Text. See *Controlling the Mapping of Large Character Data Types* for more information.
6. From the Select Application Directory drop-down list, select a directory. The default value is baseapp.
7. Complete your table or view selection:

To select all tables or views in the list, click **All**.

To select specific tables or views, click the corresponding check boxes.

8. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.
9. Click *Create Synonym*. Synonyms are created and added under the specified application directory.

Reference **Managing Synonyms**

In the navigation pane, click the name of the synonym to access the following options:

Sample Data	Retrieves up to 20 rows from the associated data source.
Properties	Displays a graphic representation of the synonym and enables you to edit its metadata.
Edit Master File	Enables you to manually edit the synonym's Master File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Master File, to update the synonym.
Edit Access File	Enables you to manually edit the synonym's Access File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Access File, to update the synonym.
Refresh	Regenerates the synonym. Use this option if structural changes were made to the data source.
Drop	Deletes the synonym.
Copy to	Copies the synonym to another application directory. Click the target directory from the resulting list.
Move to	Moves the synonym to another application directory. Click the target directory from the resulting list.

Syntax **How to Create a Synonym Manually**

```
CREATE SYNONYM app/synonym FOR table_view DBMS SQLINF [AT  
connection][NOCOLS]  
END
```

where:

app

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, then this application name must not be used.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema) as follows:

```
[owner.] table
```

SQLINF

Indicates the Data Adapter for Informix.

AT *connection*

Is the service name as previously specified in a SET CONNECTION_ATTRIBUTES command. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

This parameter is optional. If specified, the CONNECTION attribute is added to the Access File.

NOCOLS

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR "edaga".nf29004 DBMS SQLINF AT conn_inf
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION ,SUFFIX=SQLINF,$
SEGNAME=SEG1_4 ,SEGTYPE=S0 , $
FIELD=DIVISION4 ,DIVISION4 ,I9 ,I4 ,MISSING=OFF , $
FIELD=DIVISION_NA4,DIVISION_NA4 ,A25 ,A25 ,MISSING=ON , $
FIELD=DIVISION_HE4,DIVISION_HE4 ,I9 ,I4 ,MISSING=ON , $
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4, TABLENAME="edaga".nf29004,
CONNECTION=conn_inf, KEYS=1, WRITE=YES, $
```

Reference Access File Keywords

Attribute	Description
TABLENAME	Identifies the Informix table name. The table name can be fully qualified as follows: TABLENAME=[[database.]owner.]table
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION= <i>connection</i>

Data Type Support

The following table lists how the server maps Informix data types. Note that you can:

- Control the mapping of large character data types.
- Control the mapping of variable-length data types.
- Change the precision and scale of numeric columns.

Be sure to review those options carefully as they affect the mapping of data types.

Informix Data Type	Data Type		Remarks
	Usage	Actual	
CHAR (<i>n</i>)	<i>An</i>	<i>An</i>	<i>n</i> is an integer between 1 and 32767.
CHAR VARYING (<i>m,r</i>)	<i>A(m,r)</i>	<i>A(m,r)</i>	<i>m</i> is an integer between 1 and 255 (254 with index). <i>r</i> is an integer between 0 and 255, and less than <i>m</i> . 0 is the default value.
LVARCHAR	A2048	A2048	
NCHAR (<i>n</i>)	<i>An</i>	<i>An</i>	<i>n</i> is an integer between 1 and 32767.
NVARCHAR	A255	A255	
DECIMAL (<i>p,s</i>)	P33.33	P(<i>p,s</i>)	<i>p</i> is an integer between 1 and 32. The default value for <i>p</i> is 16. <i>s</i> is an integer between 0 and <i>p</i> . The default value for <i>s</i> is 0.
FLOAT (<i>n</i>)	D20.2	D8	<i>n</i> is an integer between 1 and 14.
DOUBLE PRECISION (<i>n</i>)	D20.2	D8	<i>n</i> is an integer between 1 and 14.
INT	I11	I4	
INT8	P20	P10	
MONEY(<i>p,s</i>)	P18.2	P9	<i>p</i> is an integer between 1 and 32. The default value for <i>p</i> is 16. <i>s</i> is an integer between 0 and <i>p</i> . The default value for <i>s</i> is 2.

Informix Data Type	Data Type		Remarks
	Usage	Actual	
SERIAL	I11	I4	
SERIAL8	P20	P10	
SMALLINT	I6	I4	
BYTE	BLOB	BLOB	
TEXT	A32767	A32767	
BLOB			Not supported for the current release of the server.
CLOB			Not supported for the current release of the server.
DATE	YYMD	DATE	
DATETIME	HYYMD	HYYMD	
INTERVAL YEAR (<i>year_precision</i>) TO MONTH	A20	A20	
INTERVAL DAY (<i>day_precision</i>) TO SECOND			
(<i>fractional_seconds_precision</i>)	A20	A20	
BOOLEAN	A1	A1	The only values that can be inserted are T or F.

Controlling the Mapping of Large Character Data Types

The SET parameter CONVERSION LONGCHAR controls the mapping of supported Informix data types listed below. By default, the server maps these data types as alphanumeric (A). The server data type A supports a maximum of 4096 characters for TABLE/MODIFY and 32768 characters for API applications.

The following table lists data type mappings based on the value of LONGCHAR:

Informix Data Type	Remarks	LONGCHAR ALPHA or BLOB		LONGCHAR TEXT	
		Usage	Actual	Usage	Actual
CHAR (n)	n is an integer between 1 and 32767	An	An	TX50	TX
NCHAR (n)	n is an integer between 1 and 32767	An	An	TX50	TX
LVARCHAR		A2048	A2048	TX50	TX
TEXT		A32767	A32767	TX50	TX

Syntax

How to Control the Mapping of Large Character Data Types

```
ENGINE [SQLINF] SET CONVERSION LONGCHAR {ALPHA|TEXT|BLOB}
```

where:

SQLINF

Indicates the Data Adapter for Informix. You can omit this value if you previously issued the SET SQLENGINE command.

ALPHA

Maps the Informix data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as alphanumeric (A). This value is the default.

TEXT

Maps the Informix data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as text (TX). Use this value for WebFOCUS applications.

BLOB

For UNIX, Windows, OpenVMS, and OS/400, is identical to ALPHA. That is, it maps the Informix data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as alphanumeric (A).

For OS/390 and z/OS, maps the Informix data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as binary large object (BLOB).

Controlling the Mapping of Variable-Length Data Types

The SET parameter VARCHAR controls the mapping of the Informix data types VARCHAR, VARCHAR2, and NVARCHAR2. By default, the server maps these data types as alphanumeric (A).

The following table lists data type mappings based on the value of VARCHAR:

Informix Data Type	Remarks	VARCHAR ON		VARCHAR OFF	
		Usage	Actual	Usage	Actual
LVARCHAR (n)		A2048V	A2048V	A2048	A2048
NVARCHAR (n)		A255V	A255V	A255	A255

Syntax

How to Control the Mapping of Variable-Length Data Types

```
ENGINE [SQLINF] SET VARCHAR {ON|OFF}
```

where:

SQLINF

Indicates the Data Adapter for Informix. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Maps the Informix data types VARCHAR, VARCHAR2, and NVARCHAR2 as variable-length alphanumeric (AnV).

OFF

Maps the Informix data types VARCHAR, VARCHAR2, and NVARCHAR2 as alphanumeric (A). This value is the default.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Syntax

How to Override Default Precision and Scale

```
ENGINE [SQLINF] SET CONVERSION RESET  
ENGINE [SQLINF] SET CONVERSION format RESET  
ENGINE [SQLINF] SET CONVERSION format [PRECISION pp [ss]]  
ENGINE [SQLINF] SET CONVERSION format [PRECISION MAX]
```

where:

SQLINF

Indicates the Data Adapter for Informix. You can omit this value if you previously issued the SET SQLENGINE command.

RESET

Returns any previously specified precision and scale values to the data adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

format

Is any valid format supported by the data source. Possible values are:

INTEGER which indicates that the command applies only to INT columns.

DECIMAL which indicates that the command applies only to DECIMAL columns.

FLOAT which indicates that the command applies only to double precision floating point columns.

pp

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

ss

Is the scale. This is valid with DECIMAL and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. If the scale is not required, you must set *ss* to 0 (zero).

MAX

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE SQLINF SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLINF SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE SQLINF SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLINF SET CONVERSION RESET
```

Customizing the Informix Environment

The Data Adapter for Informix provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Syntax

How to Obtain the Number of Rows Updated or Deleted

```
ENGINE [SQLINF] SET PASSRECS {ON|OFF}
```

where:

SQLINF

Indicates the Data Adapter for Informix. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. This value is the default.

OFF

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Calling an Informix Stored Procedure Using SQL Passthru

Informix stored procedures are supported via SQL Passthru. These procedures need to be developed within Informix using the CREATE PROCEDURE command.

Example Calling a Stored Informix Procedure

The supported syntax to call a stored procedure is shown below. It is recommended that you use the syntax below instead of the previously supported CALLINF syntax.

```
ENGINE SQLINF
EX SAMPLE PARM1, PARM2, PARM3...;
TABLE ON TABLE PCHOLD
END
```

The server supports invocation of stored procedures written according to the following rules:

- Only scalar input parameters are allowed, but not required.
- Output parameters are not allowed.

Example Informix Stored Procedure

```
CREATE PROCEDURE SAMPLE()
RETURNING CHAR(11), CHAR(20), CHAR(15), DATE, CHAR(1);
DEFINE a1 CHAR(11);
DEFINE b1 CHAR(20);
DEFINE c1 CHAR(15);
DEFINE d1 DATE;
DEFINE e1 CHAR(1);
FOREACH entry FOR SELECT ssn5, last_name5, first_name5, birthdate5, sex5
INTO
a1,b1,c1,d1,e1 FROM 'edaqa'.NF29005
RETURN a1,b1,c1,d1,e1 WITH RESUME;
CONTINUE FOREACH;
END FOREACH
END PROCEDURE;
```

The following syntax is used to call the above stored procedure:

```
SQL SQLINF
EX SAMPLE;
TABLE ON TABLE PCHOLD
END
```

Calling an Informix Stored Procedure Using SQL Passthru

CHAPTER 14

Getting Started in Ingres II

Topics:

- Preparing the Ingres II Environment
- Configuring the Data Adapter for Ingres II
- Managing Ingres II Metadata

The Data Adapter for Ingres II allows applications to access Ingres II data sources. The adapter converts application requests into native Ingres II statements and returns optimized answer sets to the requesting application.

Preparing the Ingres II Environment

Currently, the Ingres II environment is supported on UNIX and OpenVMS.

Procedure How to Set Up the Environment on UNIX

The following environmental variables must be set in this environment:

Specify the location where Ingres II was installed.

```
II_SYSTEM = /rdbms/ing250
```

Specify the installation code for Ingres II.

```
II_INSTALLATION = Ia
```

Specify the path to Ingres II shared libraries.

```
LIBPATH=/rdbms/ing250/ingres/lib
```

Procedure How to Set Up the Environment on OpenVMS

Check with the System Administrator to see if system logicals have been set up for Ingres II.

Configuring the Data Adapter for Ingres II

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

To connect to Ingres II using SET commands, issue the following:

```
ENGINE SQLING SET DATABASE dbname  
ENGINE SQLING SET USER userid/password  
ENGINE SQLING SET OPTIONS -R"xxxxxx", -G"yyyyyy"
```

where:

dbname

Is the name of the database.

userid

Is the user ID of the user.

password

Is the password in the Ingres II database. This is only required if password is turned on in the database.

xxxxxx

Is the Ingres II role ID.

yyyyyy

Is the Ingres II group ID.

Reference Declaring Ingres II Connection Attributes From the Web Console

Attribute	Description
Data source	Enter the Ingres II data source name. There is no default DSN; a value must be entered.
User	Enter the valid Ingres II username.
Password	Enter the password that identifies the entered user name.

Overriding the Default Connection

Once connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax How to Change the Default Connection

```
ENGINE [SQLING] SET DEFAULT_CONNECTION [connection]
```

where:

SQLING

Indicates the Data Adapter for Ingres II. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the service name defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following error message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following error message is issued: FOC1671, Command out of sequence.

Example Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the Ingres II database server named SAMPLENAME as the default Ingres II database server:

```
ENGINE SQLING SET DEFAULT_CONNECTION SAMPLENAME
```

Managing Ingres II Metadata

When the server accesses a data source, it needs to know how to interpret the data that it finds. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Ingres II data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Ingres II table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Editing Metadata page opens.

To create a synonym, you must have configured the adapter. See *Configuring the Data Adapter for Ingres II* for more information.

2. Expand the *Add* folder, expand the adapter folder, and then click a connection. The right pane displays table and view selection options:

Select All Tables/Views. Select this radio button to create synonyms for all tables and views. This value is the default.

Filter by Name, Owner and Table Type. Select this radio button to filter the tables or views for which to create synonyms.

Selecting this option adds the following:

Select all Tables/Views
 Filter by Name, Owner and Table Type

Owner - Sample: abc%
 Table name - Sample: abc%
 Table type TABLE VIEW TABLE/VIEW

Select Tables

Owner. Enter a string for filtering the owners' IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owners' IDs begin with the letters ABC; %ABC to select tables or views whose owners' IDs end with the letters ABC; %ABC% to select tables or views whose owners' IDs contain the letters ABC at the beginning, middle, or end.

Table Name. Enter a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

Table Type. Select one of the following options: Tables (this is the default), Views, Tables/Views.

3. Click *Select Tables*. All tables that meet the specified criteria are displayed:

Prefix: SET CONVERSION LONGCHAR: TEXT (for Web Focus) ▼

Select Application Directory: ▼

All Select

Check	Default Synonym Name	Owner/Schema	Table Name	Type
<input type="checkbox"/>	<input type="text" value="NF29001"/>	EDAQA	NF29001	TABLE
<input type="checkbox"/>	<input type="text" value="NF29002"/>	EDAQA	NF29002	TABLE
<input type="checkbox"/>	<input type="text" value="NF29003"/>	EDAQA	NF29003	TABLE
<input type="checkbox"/>	<input type="text" value="NF29004"/>	EDAQA	NF29004	TABLE
<input type="checkbox"/>	<input type="text" value="NF29005"/>	EDAQA	NF29005	TABLE
<input type="checkbox"/>	<input type="text" value="NF29006"/>	EDAQA	NF29006	TABLE
<input type="checkbox"/>	<input type="text" value="NF29007"/>	EDAQA	NF29007	TABLE
<input type="checkbox"/>	<input type="text" value="NF29008"/>	EDAQA	NF29008	TABLE
<input type="checkbox"/>	<input type="text" value="NF29009"/>	EDAQA	NF29009	TABLE
<input type="checkbox"/>	<input type="text" value="NF29012"/>	EDAQA	NF29012	TABLE
<input type="checkbox"/>	<input type="text" value="NF29013"/>	EDAQA	NF29013	TABLE

4. If you have tables with identical table names, assign a prefix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave field blank.

5. From the SET CONVERSION LONGCHAR drop-down list, select: Text, Alpha, or BLOB. The default value is Text.
6. From the Select Application Directory drop-down list, select a directory. The default value is baseapp.
7. Complete your table or view selection:
To select all tables or views in the list, click **All**.
To select specific tables or views, click the corresponding check boxes.
8. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.
9. Click *Create Synonym*. Synonyms are created and added under the specified application directory.

Reference **Managing Synonyms**

In the navigation pane, click the name of the synonym to access the following options:

Sample Data	Retrieves up to 20 rows from the associated data source.
Properties	Displays a graphic representation of the synonym and enables you to edit its metadata.
Edit Master File	Enables you to manually edit the synonym's Master File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Master File, to update the synonym.
Edit Access File	Enables you to manually edit the synonym's Access File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Access File, to update the synonym.
Refresh	Regenerates the synonym. Use this option if structural changes were made to the data source.
Drop	Deletes the synonym.
Copy to	Copies the synonym to another application directory. Click the target directory from the resulting list.
Move to	Moves the synonym to another application directory. Click the target directory from the resulting list.

Syntax **How to Create a Synonym Manually**

```
CREATE SYNONYM app/synonym FOR datasource DBMS SQLING [NOCOLS]  
END
```

where:

app

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, then this application name must not be used.

synonym

Is an alias for the data source (maximum 64 characters).

datasource

Is the fully qualified name for the physical data structure.

SQLING

Indicates the Data Adapter for Ingres II.

NOCOLS

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.NF29004 DBMS SQLING AT DSN_A
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION ,SUFFIX=SQLING,$
SEGNAME=SEG1_4 ,SEGTYPE=S0 ,$
FIELD=DIVISION4 ,DIVISION4 ,I9 ,I4 ,MISSING=OFF ,$
FIELD=DIVISION_NA4,DIVISION_NA4 ,A25 ,A25 ,MISSING=ON ,$
FIELD=DIVISION_HE4,DIVISION_HE4 ,I9 ,I4 ,MISSING=ON ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4, TABLENAME=EDAQA.NF29004,
CONNECTION=DSN_A, KEYS=1, WRITE=YES, $
```

Data Type Support

The following chart provides information about the default mapping of Ingres II data types to server data types:

Ingres II Data Type	Data Type		Remarks
	Usage	Actual	
CHAR (<i>n</i>)	A(256)	A(256)	
VARCHAR (<i>n</i>)	A(256)	A(256)	
LONG VARCHAR			not supported
TEXT A(254) A(254)	A(254)	A(254)	
DATE	HYYMDS	HYYMDS	
DATETIME			not supported
MONEY	P18.2	P18.2	
FLOAT	D20.2	D20.2	
INTEGER	I11	I4	
SMALLINT	I6	I4	
DECIMAL(<i>p,s</i>)	P11	P4	
BYTE			not supported
BYTE VARYING			not supported
LONG BYTE			not supported

CHAPTER 15

Getting Started in Interplex

Topics:

- Preparing the Interplex Environment
- Configuring the Data Adapter for Interplex
- Managing Interplex Metadata
- Customizing the Interplex Environment

The Data Adapter for Interplex allows applications to access Interplex data sources. The adapter converts application requests into native Interplex statements and returns optimized answer sets to the requesting application.

Preparing the Interplex Environment

Prior to configuring the Interplex Data Adapter, the following must be installed:

A middleware product on the Unisys platform.

- An ODBC driver.

Configuring the Data Adapter for Interplex

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to an Interplex database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Manually add the command in the global server profile (edasprof.prf) or in a user profile (user.prf).
- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the global server profile. For more information, see *Declaring Interplex Connection Attributes From the Web Console*.

You can declare connections to more than one Interplex database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to Interplex Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Syntax How to Declare Connection Attributes Manually

```
ENGINE [SQLIPX] SET CONNECTION_ATTRIBUTES [data source]/user_ID,password
```

where:

SQLIPX

Indicates the Data Adapter for Interplex. You can omit this value if you previously issued the SET SQLENGINE command.

data_source

Is the name of the Interplex data source you wish to access.

user_ID

Is the primary authorization ID by which you are known to Interplex.

password

Is the password associated with the primary authorization ID.

Example Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to the Interplex database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLIPX SET CONNECTION_ATTRIBUTES SAMPLESERVER/MYUSER,PASS
```

Reference Declaring Interplex Connection Attributes From the Web Console

Attribute	Description
Data source	Enter the data source name configured using the Interplex Driver Manager.
User	Enter the primary authorization ID by which you are known to Interplex data source.
Password	Enter the password associated with the primary authorization ID.

Overriding the Default Connection

Once connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax How to Change the Default Connection

```
ENGINE [SQLIPX] SET DEFAULT_CONNECTION [connection]
```

where:

SQLIPX

Indicates the Data Adapter for Interplex. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the service name defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following error message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following error message is issued: FOC1671, Command out of sequence.

Example Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the Interplex database server named SAMPLENAME as the default Interplex database server:

```
ENGINE SQLIPX SET DEFAULT_CONNECTION SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax How to Control the Connection Scope

```
ENGINE [SQLIPX] SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

SQLIPX

Indicates the Data Adapter for Interplex. You can omit this value if you previously issued the SET SQLENGINE command.

FIN

Disconnects automatically only after the session has been terminated. This value is the default.

COMMIT

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Interplex Metadata

When the server accesses a data source, it needs to know how to interpret the data that it finds. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Interplex data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Interplex table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Editing Metadata page opens.

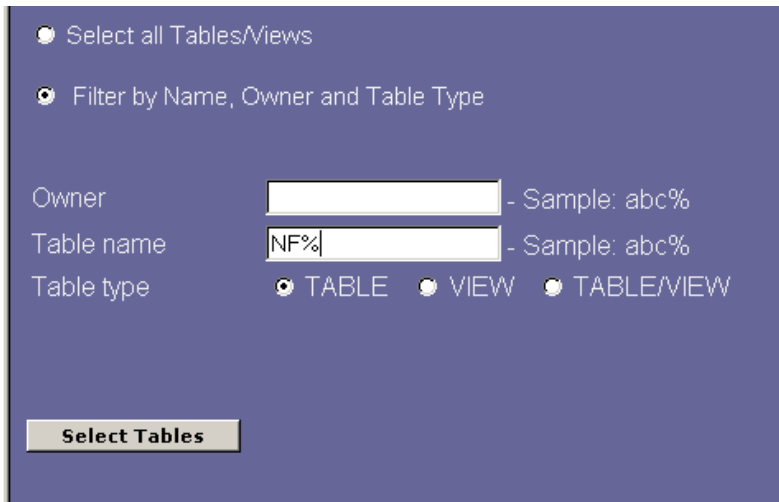
To create a synonym, you must have configured the adapter. See *Configuring the Data Adapter for Interplex* for more information.

2. Expand the *Add* folder, expand the adapter folder, and then click a connection. The right pane displays table and view selection options:

Select All Tables/Views. Select this radio button to create synonyms for all tables and views. This value is the default.

Filter by Name, Owner and Table Type. Select this radio button to filter the tables or views for which to create synonyms.

Selecting this option adds the following:



Select all Tables/Views

Filter by Name, Owner and Table Type

Owner - Sample: abc%

Table name - Sample: abc%

Table type TABLE VIEW TABLE/VIEW

Select Tables

Owner. Enter a string for filtering the owners' IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owners' IDs begin with the letters ABC; %ABC to select tables or views whose owners' IDs end with the letters ABC; %ABC% to select tables or views whose owners' IDs contain the letters ABC at the beginning, middle, or end.

Table Name. Enter a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

Table Type. Select one of the following options: Tables (this is the default), Views, Tables/Views.

3. Click *Select Tables*. All tables that meet the specified criteria are displayed:

Prefix: SET CONVERSION LONGCHAR: TEXT (for Web Focus) ▼

Select Application Directory: baseapp ▼

All Select

Check	Default Synonym Name	Owner/Schema	Table Name	Type
<input type="checkbox"/>	<input type="text" value="NF29001"/>	EDAQA	NF29001	TABLE
<input type="checkbox"/>	<input type="text" value="NF29002"/>	EDAQA	NF29002	TABLE
<input type="checkbox"/>	<input type="text" value="NF29003"/>	EDAQA	NF29003	TABLE
<input type="checkbox"/>	<input type="text" value="NF29004"/>	EDAQA	NF29004	TABLE
<input type="checkbox"/>	<input type="text" value="NF29005"/>	EDAQA	NF29005	TABLE
<input type="checkbox"/>	<input type="text" value="NF29006"/>	EDAQA	NF29006	TABLE
<input type="checkbox"/>	<input type="text" value="NF29007"/>	EDAQA	NF29007	TABLE
<input type="checkbox"/>	<input type="text" value="NF29008"/>	EDAQA	NF29008	TABLE
<input type="checkbox"/>	<input type="text" value="NF29009"/>	EDAQA	NF29009	TABLE
<input type="checkbox"/>	<input type="text" value="NF29012"/>	EDAQA	NF29012	TABLE
<input type="checkbox"/>	<input type="text" value="NF29013"/>	EDAQA	NF29013	TABLE

4. If you have tables with identical table names, assign a prefix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave field blank.

5. From the SET CONVERSION LONGCHAR drop-down list, select: Text, Alpha, or BLOB. The default value is Text.
6. From the Select Application Directory drop-down list, select a directory. The default value is baseapp.
7. Complete your table or view selection:

To select all tables or views in the list, click **All**.

To select specific tables or views, click the corresponding check boxes.

8. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.
9. Click *Create Synonym*. Synonyms are created and added under the specified application directory.

Reference **Managing Synonyms**

In the navigation pane, click the name of the synonym to access the following options:

Sample Data	Retrieves up to 20 rows from the associated data source.
Properties	Displays a graphic representation of the synonym and enables you to edit its metadata.
Edit Master File	Enables you to manually edit the synonym's Master File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Master File, to update the synonym.
Edit Access File	Enables you to manually edit the synonym's Access File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Access File, to update the synonym.
Refresh	Regenerates the synonym. Use this option if structural changes were made to the data source.
Drop	Deletes the synonym.
Copy to	Copies the synonym to another application directory. Click the target directory from the resulting list.
Move to	Moves the synonym to another application directory. Click the target directory from the resulting list.

Syntax **How to Create a Synonym Manually**

```
CREATE SYNONYM app/synonym FOR datasource DBMS SQLIPX AT
connection[NOCOLS]
END
```

where:

app

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, then this application name must not be used.

synonym

Is an alias for the data source (maximum 64 characters).

datasource

Is the name of the ODBC data source you wish to access. It must match an entry in the ODBC Driver manager that is pointing to the ODBC compliant data source or table.

SQLIPX

Indicates the Data Adapter for Interplex.

AT *connection*

Is the service name as previously specified in a SET CONNECTION_ATTRIBUTES command. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

This parameter is optional. If specified, the CONNECTION attribute is added to the Access File.

NOCOLS

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.NF29004 DBMS SQLIPX AT DSN_A
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION ,SUFFIX=SQLIPX,$
SEGNAME=SEG1_4 ,SEGTYPE=S0 ,$
FIELD=DIVISION4 ,DIVISION4 ,I9 ,I4 ,MISSING=OFF ,$
FIELD=DIVISION_NA4 ,DIVISION_NA4 ,A25 ,A25 ,MISSING=ON ,$
FIELD=DIVISION_HE4 ,DIVISION_HE4 ,I9 ,I4 ,MISSING=ON ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4 ,TABLENAME=EDAQA.NF29004 ,
CONNECTION=DSN_A ,KEYS=1 ,WRITE=YES , $
```

Reference Access File Keywords

Keyword	Description
TABLENAME	Identifies the Interplex table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: TABLENAME=[owner.]table [@databaselink]
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=connection CONNECTION='' indicates access to the local Interplex database server. Absence of the CONNECTION attribute indicates access to the default database server.

Data Type Support

Data types are specific to the underlying data source.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Syntax

How to Override Default Precision and Scale

```
ENGINE [SQLIPX] SET CONVERSION RESET
ENGINE [SQLIPX] SET CONVERSION format RESET
ENGINE [SQLIPX] SET CONVERSION format [PRECISION pp [ss]]
ENGINE [SQLIPX] SET CONVERSION format [PRECISION MAX]
```

where:

SQLIPX

Indicates the Data Adapter for Interplex. You can omit this value if you previously issued the SET SQLENGINE command.

RESET

Returns any previously specified precision and scale values to the data adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

format

Is any valid format supported by the data source. Possible values are:

INTEGER which indicates that the command applies only to INT columns.

DECIMAL which indicates that the command applies only to DECIMAL columns.

FLOAT which indicates that the command applies only to double precision floating point columns.

pp

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

ss

Is the scale. This is valid with DECIMAL and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. If the scale is not required, you must set *ss* to 0 (zero).

MAX

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE SQLIPX SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLIPX SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE SQLIPX SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLIPX SET CONVERSION RESET
```

Customizing the Interplex Environment

The Data Adapter for Interplex provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Syntax **How to Obtain the Number of Rows Updated or Deleted**

```
ENGINE [SQLIPX] SET PASSRECS {ON|OFF}
```

where:

SQLIPX

Indicates the Data Adapter for Interplex. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. This value is the default.

OFF

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

CHAPTER 16

Getting Started in J.D. Edwards Query Adapter

Topics:

- Overview of the Setup Process
- Installation Prerequisites
- Unloading the CD-ROM
- Working With QSYS and IFS File Systems
- Generating Metadata
- Using Master Files and Access Files
- Enabling J.D. Edwards Security
- Using the Report Library
- Enabling Tracing
- J.D. Edward Major System Codes
- Frequently Asked Questions

The Data Adapter for J.D. Edwards allows WebFOCUS and other applications to access J.D. Edwards data sources. The adapter converts data or application requests into native J.D. Edwards statements and returns optimized answer sets to the requesting program.

Overview of the Setup Process

To enable access to J.D. Edwards, perform the following steps:

1. Unload the I52TOOLS library from the CD-ROM.
2. Run either AUTOSQL or AUTO400 to generate metadata for the J.D. Edwards data.
Note: You will use the Web Console to generate the metadata for J.D. Edwards data because the metadata generated by the Web Console differs in the use of long field names and extra attributes such as title and description. You must use AUTOSQL or AUTO400 to generate your metadata.
3. Run JDECONV to update the metadata with information derived from the J.D. Edwards Data Dictionary.
4. Enable J.D. Edwards security on the server.

Installation Prerequisites

The software requirements for the Data Adapter for J.D. Edwards are:

- OS/400 version V4R4 or higher on a RISC system.
- iWay server release 5.2, or higher, should be installed and working. See the *iWay Server Installation* manual.

Note: If the standard IADMIN ID was not used to install the iWay Server, manually change ownership to the ID that was used.

You must also verify that your end-user program (such as WebFOCUS Desktop) can access the iWay server.

Unloading the CD-ROM

Note: The I52TOOLS library on the CD-ROM is a limited packaging of specific tools from what was the EDA43TOOLS library from the 4.x Release. The I51TOOLS and the I52TOOLS CD-ROM differ only in the OS/400 library name and the version of the documentation included on the CD-ROM. If you already have an EDA43TOOLS or I51TOOLS library installed, you may continue to use them and these recommendations. You do not need to install the new library.

To unload the CD-ROM, perform the following steps:

1. If *SECOFR Authority is required to access the CD-ROM drive, logon as QSECOFR or with a user ID that has this authority. Otherwise, logon using the IADMIN ID.
2. Load the CD-ROM into the CD-ROM drive on your system.
3. When the drive is ready, use the RSTLIB command to restore the I52TOOLS library. From the command line, the syntax is

```
RSTLIB SAVLIB(I52TOOLS) DEV(device-name)
```

where:

device-name

Is the name of the CD-ROM device. For a list of valid drives, use the [WRKCFGSTS *DEV](#) command.

When unload is complete, you are returned to the command line and the library is ready to use.

End users requiring the I52TOOLS can add the library to the path with:

```
ADDLIBLE I52TOOLS
```

Working With QSYS and IFS File Systems

QSYS is the original library-based file system of the OS/400 operating system. IFS is a UNIX-like file system consisting of a directory hierarchy and files within the hierarchy; these are also known as *directories* when working with native OS/400 commands. Along with IFS is a UNIX command shell environment that is activated with the QSH command.

To use QSYS members with iWay 5.x servers, you can copy or link files from QSYS to IFS, or you can set EDAPATH to *see* QSYS libraries.

The 4.x release of the iWay server was an OS/400 product using strictly QSYS libraries for the storage of the actual product files and end-user applications. The 5.x release of the server uses a QSYS library (EDAHOMELIB) for the actual product programs and IFS directories and files for other product files and end-user applications. The EDAPATH and APP MAP methods enable you to access existing applications in QSYS libraries without physically moving them from one file system to another.

Since the iWay tools were originally designed to work with a QSYS-based system, they strictly use QSYS libraries as read and write locations. The strict use of QSYS for these tools may necessitate manual steps to copy files between QSYS and IFS depending upon your application needs. The command syntax for this type of copy is documented in this guide and can be scripted if many files are involved.

Copying Files

Files may be copied to and from QSYS and IFS, but one must be careful which copy is being accessed at any given time in case edits were done to one copy but not the other. To avoid confusion, be sure to clean up unused files.

The following example shows how to copy a Master File from one location to the other:

```
CPYTOSTMF FROMMBR('/qsys.lib/myfoo.lib/master.file/foo.mbr')
          TOSTMF('/home/iadmin/ibi/srv52/ffs/catalog/foo.mas')
          STMFOPT(*REPLACE)

CPYFRMSTMF FROMSTMF('/home/iadmin/ibi/srv52/ffs/catalog/foo.mas ')
          TOMBR('/qsys.lib/myfoo.lib/master.file/foo.mbr')
          MBROPT(*REPLACE)
```

Note how each command works only in a single directory and uses a UNIX-like construct, even for the QSYS name. Use the *REPLACE option to overwrite an existing file.

To copy an Access File, the file name and extension are: focsql and .acx. To copy a stored procedure (focexec), the file name and extension are: focexec and .fex.

CPYTOSTMF and CPYFRMSTMF are OS/400 commands provided by the operation system; they support the use of PF4 for interactive use.

Linking Files

IFS supports the creation of symbolic links between the QSYS file system and the IFS file system. A symbolic link allows the same file to be visible from both file systems. Symbolic links work only when the QSYS target is the real file and the IFS target is the virtual file. It may be thought of as a link that can be created in only one direction: QSYS to IFS. From an application perspective, this allows a file to co-exist in QSYS and IFS. If the file originated in IFS then setting up a link would require copying to QSYS, removing the IFS version and then creating the link.

Symbolic links are created from the QSH environment as shown in the following example for a Master File:

```
ln -s /qsys.lib/myfoo.lib/master.file/foo.mbr \
    /home/iadmin/ibi/srv52/ffs/catalog/foo.mas
```

To create a link for an Access File, the file name and extension are: focsql and .acx. To create a link for a stored procedure (focexec), the file name and extension are: focexec and .fex.

Using EDAPATH

EDAPATH, long used on other server platforms, was introduced to the iWay Server for OS/400 in Release 5.1. As of Release 5.2, path searches use the APP method, which was also available in Release 5.1, but was not the default method. In Release 5.2, you can continue to use EDAPATH as documented below if APP is disabled (in the Web Console), but it is advised that you switch to the APP method as describe in *Using APP* on page 16-6.

EDAPATH is the search path for iWay related applications such as focexecs (.fex), Masters (.mas), Access files (.acx), and FOCUS databases. In an IFS context, EDAPATH may be set to one or more directories, separated by colons. In a QSYS context, EDAPATH may be set to one or more IFS references for QSYS libraries, separated by colons. The EDAPATH value may be set to any number of intermixed IFS or QSYS references, separated by colons.

Path search against EDAPATH for QSYS library references makes the standard EDA 3.x and 4.x application library members of MASTER, FOCSQL and FOCEXEC available for use by iWay 5.x servers. Thus any of the tools documented here (that write files to libraries) may continue to be used without the need to copy or link files, provided EDAPATH is set appropriately.

For instance, if there were accounting and shipping libraries respectively named ACCTNG and SHIP libraries from applications built with release 4.3.1, you would access them by issuing the following:

```
SET EDAPATH = /QSYS.LIB/ACCTNG.LIB : /QSYS.LIB/SHIP.LIB
```

If SHIP applications were moved to IFS, then the specification would be:

```
SET EDAPATH = /QSYS.LIB/ACCTNG.LIB : /home/iadmin/ship
```

EDAPATH is typically set as a command in the global server profile, edasprof.prf. It may also be set via a remote procedure call, or as an environment variable before server start up; this is done by using WRKENVVAR or QSH export of the variable to the list of desired (colon-separated) references.

Using APP

APP has a feature called APP MAP which allows customers to create aliases to directories outside the APPROOT location so that they can then be used in an APP PATH, APP PREPENDPATH or APP APPENDPATH command. An actual map does not automatically place the location on the path, it is a two step process. Below is how it would be coded in the edasprof.prf, however, these names and values can also be set using the Web Console.

```
APP MAP ACCTNG /QSYS.LIB/ACCTNG.LIB
APP MAP SHIP /QSYS.LIB/SHIP.LIB
APP PATH QMETADATA SHIP IBISAMP
```

If one of the applications was moved to IFS, then the specification would be:

```
APP MAP ACCTNG /QSYS.LIB/ACCTNG.LIB
APP PATH QMETADATA SHIP IBISAMP
```

Generating Metadata

Generating metadata for J.D. Edwards is a two-step process:

1. Generate Master and Access Files using either AUTOSQL, AUTO400, or the Web Console.
2. Use the JDECONV utility to further configure the Master and Access Files.

Reminder: iWay metadata is made up of a Master File and an optional Access File.

Before using any of the tools, verify that your library list includes tools library and either QTEMP or a writable CURLIB library (use DSPLIBL to view and EDTLIBL to change).

When creating metadata, the auto facilities automatically create MASTER and FOCSQL files and members as needed in the selected target library if they do not exist. The library and files may also be pre-created using the following commands:

```
CRTLIB library-name
CRTSRCPF library-name/MASTER RCDLEN(92)
CRTSRCPF library-name/FOCSQL RCDLEN(92)
```

AUTOSQL generates members to MASTER and FOCSQL where AUTO400 only generates members to MASTER as detailed in the respective sections. JDECONV works against existing members in MASTER.

Using AUTOSQL to Generate Master and Access Files

AUTOSQL is the automated tool for describing Master File and Access Files for DB2/400 tables and accesses the data via OS/400's SQL facility. This tool creates a Master and Access File for each specified target (wildcard specifications are allowed) as a member (of the same name), respectively, in the source physical files MASTER and FOCSQL of the specified library.

For an iWay 5.x server to access the resulting descriptions from AUTOSQL, the file(s) may be copied or symbolic linked to a directory on the EDAPATH or APP PATH, or EDAPATH may be set to see the QSYS library.

The adapter suffix in the created Master File member is SQL400 (which is an alias to DB2) and indicates to use the SQL facility for underlying data access. Files that use SUFFIX=SQL400 use an access file to indicate the physical location.

To run AUTOSQL (it is assumed that library I52TOOLS has been added to your library list), at the command line enter the following command and press Enter:

[AUTOSQL](#)

The following menu appears:

```

AUTOSQL - CREATES MASTER AND ACCESS FILE DESCRIPTIONS FOR SQL ACCESS

Type choices, press Enter

Input data file . . . . . _____ Name, generic*, *ALL, *LIST
Input data file library . . . . . _____ Name, *LIBL, *CURLIB
Output master file library. . . . . _____ Name, *CURLIB

Create PC MFD and ACX files?          N          F=FPA, E=EDA, N=NO
Host use ONLY . . . . .                N          YES=Y, NO=N
Server name . . . . .                  EDASERVE
Date format . . . . .                  _____ YMD, YYMD, DMY, MDY, MDYY, DMY
Files to be generated . . . . .        _____ MASTER, ACCESS
Use LONG, SHORT or BOTH fieldnames?   LONG
For Copy Manager use?                  N          Y=YES, N=NO
Submit to batch?                        N          Y=YES, N=NO

NOTE: The FOCUS name is truncated to 8 characters from the file name.
      Input data file can be physical or logical.
      MASTER creates both MFDs and AFDs, ACCESS creates AFDs only.
      Date format is for SQL type date only.

F3=Exit

6/36
NUM

```

Fill in the fields as described in AUTOSQL Parameters and press *Enter*. An operation completed screen displays with a continue prompt.

To rerun AUTOSQL to describe another file, type *Y*.

To end the procedure, type *N*.

Reference **AUTOSQL Parameters**

Input Field	Option	Description
Input data file Required, no default		Specifies the files that AUTOSQL will use to create Master and Access Files. You can specify these files in one of four ways:
	Name	Enter the name of a single file to create one Master File and one Access File. A Master File member will be created in the first writable MASTER file in your library list using the file name. Also, an Access File member will be created in the first writable FOCSQL file in your library list using the file name.
	Generic*	To create a subset of Master Files and Access Files that begin with the same set of generic characters, such as AB* or F09* or A1231* Only those files beginning with the generic prefix will have Master File and Access File metadata created.
	*ALL	All files in the designated library will have Master File and Access File metadata created.
	*LISTname	To use a list of files to be converted, first create that list of file names using STRSEU. That file must be created as a member of the file IBILIST somewhere in your library list. (IBILIST is a standard source physical file with a record length of 92). To convert that list, place the list name – the name of the member – preceded by an asterisk – in this field. For instance, if you created a list called MYLIST as a member of IBILIST, you would enter *MYLIST in this field.
Input data file library Required, no default.	Name	Specifies the library containing the data files to create the Master File and Access File metadata against.

Input Field	Option	Description
Output master file library Required, no default	Name	Specifies the destination library that will contain the Master Files that are created by the metadata process. This library must exist; the utility will not create it for you. You may use the following OS/400 command to create the library: <pre>CRTLIB libraryname</pre> The Master Files will be created using the source file name as members in the file MASTER. Note that the MASTER file will be created if it is not there.
Create PC MFD and ACX files? default is N	F	Creates SUFFIX =FPA Master Files for use on the PC with the FOCUS Personal Agent Server. These additional Masters are placed in the file FPAMAS in the same library as the OS/400 Master Files.
	E	Creates SUFFIX =EDA Master Files for use with PC WebFOCUS Desktop. These additional Masters are placed in the file EDAMAS in the same library as the OS/400 Master Files.
	N	Does not create Master Files for PC use.
Host use ONLY default is N	Y	Yes. The Master File will not be used for access from a client-server environment such as WebFOCUS Developer Studio.
	N	No. The Master File will be used for access from a client-server environment such as WebFOCUS Developer Studio.
Server Name No default		Inserts the name of the Server on OS/400 that will provide access to the data files. This value is only used for the SUFFIX=EDA Access File for PC access.
Date format Required, no default	YMD YYMD DMY MDY MDYY DMY	Inserts one of these supported DATE formats. This format will be used only if the field is described as a DATE field type to the OS/400. If that is the case, the USAGE will become the value selected here, and the ACTUAL will be DATE.

Input Field	Option	Description
Files to be generated Required, no default	MASTER	Creates Master (MASTER) Files and Access (FOCSQL) Files for each metadata creation.
	ACCESS	Creates Access (FOCSQL) files only for each metadata creation. Use this option when data files have been moved in your system and you want to re-write the access files versus manually editing.
Use LONG, SHORT or BOTH fieldnames? default is LONG	LONG	For the MASTER file FIELDNAME, uses the DSPFFD Column Heading of the field, if it exists else use the DSPFFD field "Field". Always use "Field" as the ALIAS in the Master File. For the EDAMAS or FPAMAS file creations ALIAS always uses value of FIELDNAME so they match. DSPFFD Column Headings with blanks or special characters get filtered into underscores in the resulting field.
	SHORT	For the MASTER file FIELDNAME and ALIAS always uses DSPFFD field "Field" value. For the EDAMAS or FPAMAS file creations ALIAS always uses value of FIELDNAME so they match.
	BOTH	For the MASTER file FIELDNAME and ALIAS, uses the DSPFFD Column Heading of the field, if it exists. Otherwise, uses the DSPFFD field "Field". DSPFFD Column Headings with blanks or special characters get filtered into underscores in the resulting field. For the EDAMAS or FPAMAS file creations ALIAS always uses value of FIELDNAME so they match.
For Copy Manager use? default is N	Note: Copy Manager has been renamed ETL.	
	Y	Yes. Fieldnames will be truncated if they exceed the maximum of 30 characters.
	N	No. Fieldnames are not adjusted for length.

Input Field	Option	Description
Submit to batch? default is N	Y	Yes. Runs this process in background in the QBATCH subsystem.
	N	No. Runs this process in foreground.

Using AUTO400 to Generate Master and Access Files

AUTO400 is the automated tool for describing Master Files for DB2/400 tables and accesses the data via OS/400's OPNQRYF facility. This tool creates a single Master File for each specified target (wildcard specifications are allowed) as a member (of the same name) in the source physical file MASTER of the specified library.

For an iWay 5.x server to access the resulting descriptions from AUTO400, the file(s) may be copied or symbolic linked to a directory on the EDAPATH or APP PATH, or EDAPATH may be set to see the QSYS library.

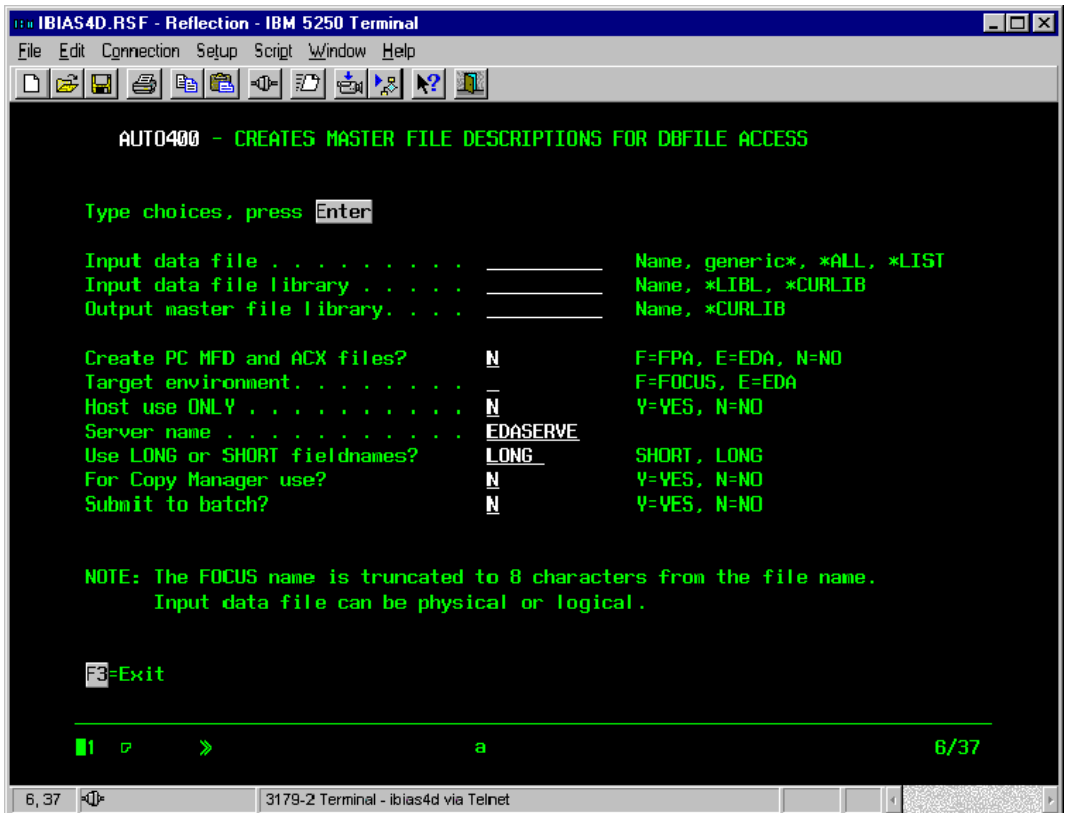
The adapter suffix in the created member is DBFILE and indicates to use the OPNQRYF facility for underlying data access. Files that use SUFFIX=DBFILE also require a FILEDEF to specify the physical location and may be issued either in an application remote procedure call or in the server's edasprof.prf. The syntax for the FILEDEF is:

```
FILEDEF AR DISK ACCTNG/AR
```

To run AUTOSQL (it is assumed that library I52TOOLS has been added to your library list), at the command line enter the following command and press Enter:

```
AUTO400
```

The following menu appears:



Fill in the fields as described in AUTO400 Parameters and press *Enter*. An operation completed screen displays with a continue prompt.

To rerun AUTO400 to describe another file, type *Y*.

To end the procedure, type *N*.

Reference AUTO400 Parameters

Input Field	Option	Description
Input data file Required, no default	Specifies the files that AUTOSQL will use to create Master and Access Files. You can specify these files in one of four ways:	
	Name	Enter the name of a single file to create one Master File and one Access File. A Master File member will be created in the first writable MASTER file in your library list using the file name. Also, an Access File member will be created in the first writable FOCSQL file in your library list using the file name.
	Generic*	To create a subset of Master Files and Access Files that begin with the same set of generic characters, such as <code>AB*</code> or <code>F09*</code> or <code>A1231*</code> Only those files beginning with the generic prefix will have Master File and Access File metadata created.
	*ALL	All files in the designated library will have Master File and Access File metadata created.
	*LISTname	To use a list of files to be converted, first create that list of file names using STRSEU. That file must be created as a member of the file IBILIST somewhere in your library list. (IBILIST is a standard source physical file with a record length of 92). To convert that list, place the list name – the name of the member – preceded by an asterisk – in this field. For instance, if you created a list called MYLIST as a member of IBILIST, you would enter *MYLIST in this field.
Input data file library Required, no default.	Name	Specifies the library containing the data files to create the Master File and Access File metadata against.

Input Field	Option	Description
Output master file library Required, no default	Name	Specifies the destination library that will contain the Master Files that are created by the metadata process. This library must exist; the utility will not create it for you. You may use the following OS/400 command to create the library: <i>CRTLIB libraryname</i> The Master Files will be created using the source file name as members in the file MASTER. Note that the MASTER file will be created if it is not there.
Create PC MFD and ACX files? default is N	F	Creates SUFFIX =FPA Master Files for use with WebFOCUS Developer Studio. These additional Masters are placed in the file FPAMAS in the same library as the OS/400 Master Files.
	E	Creates SUFFIX =EDA Master Files for use with WebFOCUS Developer Studio. These additional Masters are placed in the file EDAMAS in the same library as the OS/400 Master Files.
	N	Does not create Master Files for PC use.
Host use ONLY default is N	Y	Yes. The Master File will not be used for access from a client-server environment such as WebFOCUS Developer Studio.
	N	No. The Master File will be used for access from a client-server environment such as WebFOCUS Developer Studio.
Target environment	F	Creates Master Files specifically for use with FOCUS. Specifically, creates Master Files with Group attribute RECFORM when describing Multi-Record Format Logical files.
	E	Creates Master Files specifically for use with iWay. Specifically, creates Master Files with a Field=RECFORM when describing Multi-Record Format Logical files.
	N	No. The Master File will be used for access from a client-server environment such as WebFOCUS Desktop.

Input Field	Option	Description
Server Name No default		Inserts the name of the Server on OS/400 that will provide access to the data files. This value is only used for the SUFFIX=EDA Access File for PC access.
Date format Required, no default	YMD YYMD DMY MDY MDYY DMYY	Inserts one of these supported DATE formats (). This format will be used only if the field is described as a DATE field type to the OS/400. If that is the case, the USAGE will become the value selected here, and the ACTUAL will be DATE.
Files to be generated Required, no default	MASTER	Creates Master (MASTER) Files and Access (FOCSQL) Files for each metadata creation.
	ACCESS	Creates Access (FOCSQL) files only for each metadata creation. Use this option when data files have been moved in your system and you want to re-write the access files versus manually editing.
Use LONG or SHORT fieldnames? default is LONG	LONG	For the MASTER file FIELDNAME, uses the Row Description of the field, if it exists; otherwise, uses the Field name. Uses the Field name as the ALIAS in the Master File. For the EDAMAS or FPAMAS file FIELDNAME, uses the Row Description of the field if it exists; otherwise uses the Field Name. The ALIAS should be the same as the FIELDNAME. The ALIAS of the EDAMAS must match the FIELDNAME of the MASTER file on the host.
	SHORT	For the MASTER file FIELDNAME, uses the field name of the field. Also uses the Field name as the ALIAS in the Master File. For the EDAMAS or FPAMAS file FIELDNAME, uses the field name of the field. The ALIAS should be the same as the FIELDNAME. The ALIAS of the EDAMAS must match the FIELDNAME of the MASTER file on the host.
For Copy Manager use? default is N	Note: Copy Manager has been renamed ETL.	
	Y	Yes. Fieldnames will be truncated if they exceed the maximum of 30 characters.
	N	No. Fieldnames are not adjusted for length.

Input Field	Option	Description
Submit to batch? default is N	Y	Yes. Runs this process in background in the QBATCH subsystem.
	N	No. Runs this process in foreground.

Using JDECONV

If you will be accessing J.D. Edwards data, you will need to use the JDECONV utility to apply metadata changes to the file members built by AUTOSQL or AUTO400 tools. JDECONV updates the members (Master Files) with the metadata information dynamically derived from the J.D. Edwards Data Dictionary files. If JDECONV is not run, you will not be able to exploit any changes to the data found in the J.D. Edwards Data Dictionary.

JDECONV can only locate existing Master Files in a library's MASTER file.

For an iWay 5.x server to access the resulting descriptions converted by JDECONV, the file(s) may be copied or symbolic linked to a directory on the EDAPATH or APP PATH, or EDAPATH may be set to see the QSYS library.

The J.D. Edwards Data Dictionary Library must be present in your library list before running the JDECONV utility (use EDTLIBLE or ADDLIBLE to add). To run JDECONV, (it is assumed that library I52TOOLS has been added to your library list), at the command line enter the following command and press Enter:

[JDECONV](#)

If the J.D. Edwards library is not in the library path a message will display at the bottom of the screen (as shown below).

The following menu appears:

```

IBM ibias4j.rsf - Reflection - IBM 5250 Terminal
File Edit Connection Setup Script Window Help

JDECONV - UPDATES MFDs WITH J. D. EDWARDS DICTIONARY DATA
          SNAPack for J. D. Edwards Conversion Utility

Type choices, press Enter

Input MFDs . . . . . *ALL _____ Name, *ALL, *AUTOOUT, *LIST
Input/Output library . . . . . _____ Name
J. D. Edwards software? . . . . . W 1=OneWorld, W=World
J. D. Edwards version? . . . . . Z 6, 7, 8
Smart Dates? . . . . . Y Y=YES, N=NO
Date format . . . . . YYMD YYMD, DMY, MDYY, YYMTRD,
                               MtrDYYu,YYQ, ...
Enable Presumptive Joins? . . . . . N Y=YES, N=NO
Version of EDA? . . . . . 4.3 Valid version 3.2 or 4.3
Create PC MFD and ACX files? . . . . . N F=FPA, E=EDA, N=NO
Submit to batch? . . . . . Y Y=YES, N=NO
Combined UDC descriptions? . . . . . Y Y=YES, N=NO
For Copy Manager use? . . . . . N Y=YES, N=NO
Language code . . . . . *** Code, ***

NOTE: You must have Files F0004 F0005 and F9201 (or F9210 for JDE v8)
      in Libraries List

F3=Exit

3  a 6/36
6,36 NUM
IBM 5250 Terminal - ibias4j via Telnet
  
```

Fill in the fields as described in JDECONV Parameters and press *Enter*. An operation completed screen displays with a continue prompt.

To rerun JDECONV for another file, type *Y*.

To end the procedure, type *N*.

Reference JDECONV Parameters

Input Field	Option	Description
Input MFDs Required, no default		Specifies the files that JDECONV will operate on. You can specify a file in one of four ways:
	Name	To convert a single Master File (MFD) insert the name of a single file. The EDA Master File member will be converted and updated.
	Generic*	To create a subset of Master Files and Access Files that begin with the same set of generic characters, such as <code>AB*</code> or <code>F09*</code> or <code>A1231*</code> Only those files beginning with the generic prefix will have Master File and Access File metadata created.
	*ALL	All files in the designated library will have Master Files converted.
	*LISTname	To use a list of files to be converted, first create that list of file names using STRSEU. That file must be created as a member of the file IBILIST somewhere in you library list. (IBILIST is a standard source physical file with an record length of 92). To convert that list, place the list name – the name of the member – preceded by an asterisk – in this field. For instance, if you created a list called MYLIST as a member of IBILIST, you would insert *MYLIST in this field.
Input data file library Required, no default.	Name	Specifies the library containing the data files to create the Master File and Access File metadata against.
J.D. Edwards software default is W	1	Specifies that you are using J.D. Edwards One World software on OS/400.
	W	Specifies that you are using J.D. Edwards World software on OS/400.

Input Field	Option	Description
J.D. Edwards version default is 7	6	Specifies that you are using the 6.x versions of the J.D. Edwards World software.
	7	Specifies that you are using the 7.x versions of the J.D. Edwards World software.
	8	Specifies that you are using the 8.x versions of the J.D. Edwards One World software.
Smart Dates default is Y	Y	Yes. Applies the Date format listed below to all J.D. Edwards dates within the table(s) to be converted, using a format of P6JUL.
	N	No. Ignores the Date format listed below and displays all date formats as YMD via a DEFINE field using the GREGDT subroutine.
Date format default is YYMD	YMD YYMD DMY MDY MDYY DMYY	Inserts one of these supported DATE formats. This format will be used only if the field is described as a DATE in the J.D. Edwards Data Dictionary. If that is the case, the USAGE will become the value selected here, and the ACTUAL will be P6JUL.
Enable Presumptive JOINS default is N	Y	Yes. All fields in the MASTER file that contain J.D. Edwards Data Dictionary Items AN8, MCU and CO will have new DEFINE fields added for XREF files.
	N	No. Does not enable Presumptive JOINS.
Version of EDA default is 4.3	3.2	Specifies that you are using EDA 3.2.2 on OS/400.
	4.3	Specifies that you are using EDA 4.3.1 or iWay 5.x on OS/400.

Input Field	Option	Description
Create PC MFD and ACX files? default is N	F	Creates SUFFIX =FPA Master Files for use with WebFOCUS Developer Studio. These additional Masters are placed in the file FPAMAS6 or FPAMAS7 (depending on your J.D. Edwards software version) in the same library as the OS/400 Master Files. Use the AS4XFER utility to retrieve these files from the PC.
	E	Creates SUFFIX =EDA Master Files for use with WebFOCUS Developer Studio. These additional Masters are placed in the file EDAMAS6 or EDAMAS7 (depending on your J.D. Edwards software version) in the same library as the OS/400 Master Files. Use the SEAS4X utility to retrieve these files from the PC.
	N	Does not create Master Files for PC use.
Submit to batch default is Y	Y	Yes. Submits the program to batch.
	N	No. Submits the program to run on-line.
Combined UDC Description default is Y	Y	Yes. Creates one new DEFINE field for each UDC code. This new DEFINE field will be the combination of 2 UDC descriptions for each UDC code.
	N	No. Creates up to two new DEFINE fields for each UDC code. Each new DEFINE field will contain one of the UDC descriptions.
For Copy Manager use?	Note: Copy Manager has been renamed ETL.	
	Y	Yes. Fieldnames will be truncated if they exceed the maximum of 30 characters.
	N	No. No truncation of fieldnames will occur.

Input Field	Option	Description
Language Code default is ***	***	Determines whether you'll be using the field descriptions and titles set within J.D. Edwards data dictionary for the entered Language Code. Any field descriptions and titles set at the Major System are also used. Blank is the standard J.D. Edwards default for Domestic Language. Using the default, ***, will use the field descriptions and titles set within the DB2/400 DDS only.
	Name	Language Code value.

Using Master Files and Access Files

The following two tables describe the attributes of the Information Builders metadata (stored in Master Files and Access Files), and how it relates to J.D. Edwards metadata.

Reference Master File Attribute Sources

Attribute	Source
FILENAME	DDS File name
FIELDNAME	DDS row description/JDE F9202 table
ALIAS	DDS column name
USAGE	JDE data display rules/edit rules/JDE display decimals
ACTUAL	JDE type and length
ACCEPT	JDE acceptance criteria
TITLE	JDE column name/JDE F9202 table
DEFINE	JDE UDC's/JDE data item class
Business Unit Security	JDE MCU
Search Type Security	JDE AT1
User Defined Codes	DE F0004, F0005 and F9201 tables

Reference Access File Attribute Sources

Attribute	Source
TABlename	DDS File name
KEYS	DDS key information
KEYFLD	DDS key information
IXFLD	DDS key information
WRITE	"NO"

Enabling J.D. Edwards Security

In order to enable J.D. Edwards security, you must start your server with any of the following parameters:

`JDED=ON ;`

`JDEN=ON ;`

`JDEW=ON ;`

Note that by default, the JDED, JDEN, and JDEW parameters are OFF.

You can set these parameters either by modifying the CL source or by performing the WRKENVVAR command. For more information on the CL source, see the *iWay Server Installation* manual; for more information on WRKENVVAR, see your OS/400 documentation.

The JDED parameter, when set ON, enables automatic execution of J.D. Edwards Business Unit Security. The Server for OS/400 automatically restricts user access to data based on information retrieved from the F0001 and F0006 tables, and by then adding appropriate WHERE conditions to the users's submitted data access request. Once this has been set ON, it cannot be turned OFF until the server is shut down and then restarted (with no parameter settings).

The JDEN parameter, when set ON, enables automatic execution of J.D. Edwards Search Type Security. The Server for OS/400 server automatically restricts user access to data based on information retrieved from the F0005 table, and by then adding appropriate WHERE conditions to the users's submitted data access request. Once this has been set ON, it cannot be turned OFF until the server is shut down and then restarted (with no parameter settings).

The JDEW parameter handles column security per the F9401 table.

Using the Report Library

One of the most powerful features of the Query Adapter for J.D. Edwards is the ease with which users can customize reports. Within the reporting environments it is a simple matter to add or old columns, to change the display characteristics of report columns, to create new or different page breaks, or to create matrix reports. The list is virtually endless.

As such, the adapter contains a Report Library. The Report Library is a repository of standard J.D. Edwards reports, coded in the WebFOCUS language rather than in RPG. These reports were coded by J.D. Edwards users and have been contributed to this library of standard reports.

Please feel free to use these reports. Change them to suit your needs. You will see a dramatic reduction in coding time when changing the WebFOCUS reports rather than the standard RPG reports. This is one of the major strengths of the WebFOCUS language - productivity.

The following reports are currently included in the Report Library:

WebFOCUS Developer Studio Procedure	RPG Procedure	Report Type
F094121	P094121	General Ledger
F068515	P068515	EEO
F063002	P063002	Time/Pay

Enabling Tracing

You can easily trace any problem that arises, using the Web Console tracing facility. For more information, see the *iWay Server Administration* manual.

J.D. Edward Major System Codes

The following codes apply to J.D.Edwards World Versions 6.x, 7.x, and 8.x.

Major System Code and Name		File Prefix
00	World Foundation Environment	F00*
01	Address Book	F01*
02	Electronic Mail	F02*
03	Accounts Receivable	F03*
04	Accounts Payable	F04*
05	Stand-Alone Time Accounting	F05*
06	Payroll	F06*
07	Payroll "Enhanced"	F07*
08	Human Resources	F08*
09	General Accounting	F09*
10	Financial Reporting	F10*
11	Multi Currency/Cash Basis	F11*
12	Fixed Assets	F12*
13	Equipment/Plant Management	F13*
14	Modeling, Planning, & Budgeting	F14*
15	Commercial Property Management	F15*
16	Resident Property Management	F16*
17	Property Management Base	F17*
18	Deal Management	F18*
20	Energy Base	F19*
30	Product Data Management	F30*
31	Shop Floor Control	F31*

Major System Code and Name		File Prefix
32	Configuration Management	F32*
33	Capacity Requirements Planning	F33*
34	DRP/MRP/MPS	F34*
35	Enterprise Facility Planning	F35*
40	Inventory/OP base	F40*
41	Inventory Management	F41*
42	Sales Order Processing	F42*
43	Purchasing Order Processing	F43*
44	Contract Management	F44*
45	Advanced Price Adjustments	F45*
46	Warehouse Management	F46*
47	Electronic Data Interchange	F47*
48	Workorder Processing	F48*
49	Load and Delivery	F49*
50	Job Cost Base	F50*
51	Job Cost Accounting	F51*
52	Job Cost Billing	F52*
53	Change Management	F53*
55-59	Client Use	F55* F56* F57* F58* F59*
60-69	JDE Internal Custom Programming	F60* F61* F62* F63* F64* F65* F66* F67* F68* F69*
70	Multi-National Products	F70*
71	Client/Server Applications	F71*
72	World Vision	F72*
73	CS - A/P Entry	F73*

Major System Code and Name		File Prefix
74	CS - Pay Time Entry	F74*
75	CS - Sales Order Entry	F75*
76	CS - Training and Development	F76*
77	Canadian Payroll	F77*
79	CS - Translation	F79*
80	COBOL Translator	F80*
81	DREAM Writer	F81*
82	World Writer	F82*
83	Management Reporting - FASTR	F83*
84	Distributive Data Processing	F84*
85	Custom Programming	F85*
86	Electronic Document Interchange	F86*
87-98	Miscellaneous Tech	F87* F88* F89* F90* F91* F92* F93* F94* F95* F96* F97* F98*

Frequently Asked Questions

I ran a report against a JDE file, however, no converted precision is coming back, why?

Confirm if your request is using the short JDE field names or the long field names. If your request is using short names, then it sounds like the request is being processed via APT (Automatic Passthru), which goes directly through the SQL engine and bypasses the MASTER file format changes. To solve this issue, add the following 3 lines to the bottom of the edasprof.prf:

```
SQL
SET APT = OFF;
END
```

Remember, in iWay 5.x the edasprof.prf is in ibi/srv5*/*/etc and not in a library as done in 4.x servers.

Which library should I store my MASTER, FOCSQL and FOCEXEC files in?

We highly recommend storing all MASTER, FOCSQL and FOCEXEC files in a separate library or directory from where the server was installed so that you can add to the server's path prior to server start-up time, using either APP PATH or EDAPATH methods. This will avoid not only confusion as to where they reside, but will also make it easier to maintain during server upgrades.

I've been using the J.D. Edwards Adapter under an EDA 3.2.2 server and I'm upgrading to a newer server level. Will the MASTER files I built under EDA 3.2.2 work with EDA 4.3.1 or iWay 5.x?

No, existing MASTER files built with JDECONV from EDA 3.2.2 are not supported under EDA 4.3.1 or iWay 5.x. Enhancements have been made to the JDECONV tool for 4.3.1 to accommodate the EDA 4.3.1 architecture. Specifically, any fields for which precision is defined in the J.D. Edwards data dictionary now have a JDE suffix added to the USAGE attribute in the MASTER file. For example:

In EDA 3.2.2:

```
FIELD=BAL_FWD, ALIAS='GBAPYC', USAGE=P16.2NS, ACTUAL=P8.0,
DESCRIPTION='Bal FWD', TITLE='Beg Balance/, PYE Forward', $
```

In EDA 4.3.1 or higher:

```
FIELD=BAL_FWD, ALIAS='GBAPYC', USAGE=P16.2NS, ACTUAL=P8.0JDE,
DESCRIPTION='Bal FWD', TITLE='Beg Balance/, PYE Forward', $
```

We suggest saving your existing EDA 3.2.2 MASTER files as a backup, and re-creating the metadata with the latest tools.

CHAPTER 17

Getting Started in Lawson

Topics:

- Preparing the Server Environment for Adapter Configuration
- Configuring the Adapter for Lawson
- Managing Metadata
- Using the Query Adapter for Lawson

The Query Adapter for Lawson allows applications to access Lawson data sources.

The Query Adapter for Lawson automatically applies the appropriate Lawson security rules as defined in the Lawson LAUA repository. The Query Adapter for Lawson transparently generates dynamic DBA statements and/or dynamic WHERE clauses.

Preparing the Server Environment for Adapter Configuration

To prepare the server for adapter configuration, perform the following:

1. Extract the Lawson security information into flat files using the **lawsdmp.sh** utility.
2. Create lawsec directory and add to the path.
3. Copy the Lawson security information flat files to the server using the lawscopy utility.
4. Run the **glawfils.fex** procedure to change the flat files into a format the server can understand.

Note: It is assumed that the server is already installed.

Every time the Lawson security information is changed, you need to update the server files. See *How to Update Lawson Security Information* on page 17-6.

Procedure How to Extract the Lawson Security Information

On the machine where Lawson is running, perform the following:

1. In the Lawson System, create the ibi_officer user ID, or any user ID with Security Officer capabilities (allowed to run the Lawson RINGDBDUMP command).
2. Copy **lawsdmp.sh**, **lawsdmpu.sh**, and **lawsfil.txt** from the etc directory of EDAHOME into the home directory of the ibi_officer.

EDAHOME is the location of the files that run your server. The following table shows the default locations for EDAHOME.

Operating system	EDAHOME default directory location and name
Windows NT/2000	ibi\srv52\home
UNIX and OS/400	/home/iadmin/ibi/srv52/home

Note: For more information on EDAHOME, see the *iWay Server Administration* manual.

3. Change the execute attribute of **lawsdmp.sh** and of **lawsdmpu.sh** to *On*.
4. Schedule or run **lawsdmp.sh** (or **lawsdmpu.sh**) to extract the latest changes to the Lawson Security System.

Procedure How to Copy the Lawson Security Information Flat Files to the Server

1. Start the server and the Web Console.
2. Open the Configure Application Path window (click either *Metadata* or *Procedures* in the Web Console menu, then click *Configure Application Path*).
3. Create a directory named *lawsec* in the *APPROOT* directory by clicking *New directory*. This is the Lawson Security Directory.

APPROOT is a directory that contains applications and sample files. The following table shows the default locations for *APPROOT*.

Operating System	Default Location
Windows NT/2000	ibi\apps
UNIX, OS/400	home/iadmin/ibi/apps

Note: For more information on the Web Console and *APPROOT*, see the *iWay Server Administration* manual.

4. Include *lawsec* in the *APPPATH* by dragging the folder next to its name into the *APPPATH* box and clicking *Set APP PATH*.
5. **On UNIX**, copy **lawscopy.sh**, **lawsftp.dat**, and **lawscopy.bat** and set the execute attribute of **lawscopy.sh** to *On*.
On Windows, copy **lawsftp.dat** and **lawscopy.bat**.
6. **On UNIX**, edit **lawscopy.sh** or **lawscopy.bat**.
On Windows, edit **lawscopy.bat** to specify the correct source and target directories.
The source directory is the home directory of *ibi_officer* that you created in Step 1 of the previous section. The target directory is *lawsec*.
7. Edit **lawsftp.dat** to indicate the hostname, user ID, and password of the remote machine that contains the extracts from running the **lawsdmp.sh** script.
8. If the server with the Query Adapter for Lawson is in the same box as the Lawson System, run **lawscopy.sh** (on UNIX) or **lawscopy.bat** (on Windows) to copy the extracted files from the **lawsdmp** run to the Lawson Security Directory (the *lawsec* directory in *APPROOT*) of the server.

If not on the same box, run **ftp** with **lawsftp.dat** as input to ftp the extracted files to the Lawson Security Directory (the *lawsec* directory in *APPROOT*). This should be done as follows:

```
ftp -niv < lawsftp.dat
```

Procedure How to Run **glawfiles.fex**

1. Copy **glawcopr.fex**, **glawfiles.fex**, and **glawincl.fex (glaw*.fex)** from the catalog directory in EDAPATH to the lawsec directory.
2. If you haven't already done so, start the server and the Web Console.
3. Click *Procedures* in the Web Console menu, and then expand the lawsec directory.
4. Edit **glawincl.fex** by clicking it and clicking *Edit* in the pop-up menu.
 - a. Set EDAPATH, TEMP and &FTMDIR to the Lawson Security Directory (the lawsec directory in APPROOT).
 - b. Set &HOMECAT to the catalog directory in EDAPATH, where **lawsflds.*** resides.
 - c. Set &PLATFORM to PC if Lawson System is on Windows and to UNIX if Lawson System is on a UNIX system.

Once finished, click *Save*.

5. Run **glawfiles.fex** by clicking it, and then clicking *Run*.

This process creates the LAW* FOCUS files needed by the Query Adapter for Lawson to apply the necessary filtering to queries against the Lawson database files.

Configuring the Adapter for Lawson

Once you have completed preparing the server environment, you are ready to configure the Query Adapter for Lawson. You can configure the adapter in one of two ways:

- Using the Web Console.
- Manually editing the global server profile, edasprof.prf.

Procedure How to Configure the Query Adapter for Lawson

You can configure the adapter by doing the following in the Web Console:

1. Click *Data Adapters*.

The Configuring Data Adapters page opens.
2. If necessary, expand the Add folder.
3. Expand the Lawson folder.
4. Click the Lawson icon.
5. Enter a DBMS Suffix and a Default Product Line.
6. Click *Configure*.

Reference Declaring Lawson Connection Attributes From the Web Console

Attribute	Description
DBMS Suffix	Specify the suffix of the database that Lawson is using. Possible values are: SQLORA Oracle SQLMSS Microsoft SQL Server SQLINF Informix SQLDB2 DB2
Default Product Line	This value is customized by the site at installation of the Lawson system.

This does the following:

1. Adds the following lines to **edasprof.prf**

```
SET USER=LAWSON
ENGINE LAWSON SET LAW DBMS dbmssuffix
ENGINE LAWSON SET LAW PRODLINE lawidi
```

where:

dbmssuffix

Is the suffix for the database used by the Lawson installation.

lawidi

Is the default Lawson Product Line to access.

Note: If you change the value for SET USER, you will not be able to access Lawson data.

2. Adds the following line to **edaserve.cfg**:

```
law_access=y
```

Managing Metadata

Once the Query Adapter for Lawson is configured, set up the metadata for the underlying database management system. See the documentation for the appropriate DBMS adapter.

Using the Query Adapter for Lawson

The Query Adapter for Lawson works behind the scenes to control end users' access to the data in the Lawson system. When a user makes a request against a Lawson table, the Query Adapter for Lawson asks the following questions:

- Is this a Lawson File?
- Is this a Lawson User?
- Does the user have access to this table?

Once these questions have been answered, the Query Adapter for Lawson does the following:

- Dynamically generates DBA for any Application Security System Code, Record Level security, File Security, and Field Security. For more information on DBA, see the *Describing Data* manual.
- Generates WHERE clauses for any Lawson Company and Process Level, Record Security, and Field Security restrictions. These WHERE clauses are appended to any requests for data made to the Lawson data.

Important: Automatic Passthru has been disabled against Lawson data. You must set Automatic Passthru off with the following command:

```
SET APT = OFF
```

For more information on Automatic Passthru, see your iWay documentation.

To update the Lawson security information, you must update the server files. See *How to Update Lawson Security Information* on page 17-6.

Procedure How to Update Lawson Security Information

1. Run **lawsdmp.sh** (or **lawsdmpu.sh**) to extract the latest changes to the Lawson Security System in the ibi_officer user ID home directory.
2. If the server with the Query Adapter for Lawson is in the same box as the Lawson System, run **lawscopy.sh** (on UNIX) or **lawscopy.bat** (on Windows) to copy the extracted files from the lawsdmp run to the Lawson Security Directory (the lawsec directory in APPROOT) of the server. You can find these files in the lawsec directory.

If not on the same box, run ftp with **lawsftp.dat** as input to ftp the extracted files to the Lawson Security Directory (the lawsec directory in APPROOT). This should be done as follows:

```
ftp -niv < lawsftp.dat
```

3. Start the server and the Web Console.
4. Click *Procedures* in the Web Console menu, and then expand the lawsec directory.
5. Run **glawfils.fex** by clicking it, and then click *Run*.

CHAPTER 18

Getting Started in Microsoft Access

Topics:

- Preparing the Microsoft Access Environment
- Configuring the Data Adapter for Microsoft Access
- Managing Microsoft Access Metadata
- Customizing the Microsoft Access Environment

The Data Adapter for Microsoft Access allows applications to access Microsoft Access data sources. The adapter converts application requests into native Microsoft Access statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

Preparing the Microsoft Access Environment

The Microsoft Access Data Adapter minimally requires the installation of ODBC 32-bit Driver Manager. The configuration of a system data source name allows you to connect to a local or remote Microsoft Access data source.

Procedure How to Set Up the Environment on Windows NT/2000

There is no environment set up needed to access Microsoft Access.

Configuring the Data Adapter for Microsoft Access

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to an Microsoft Access database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Manually add the command in the global server profile (edasprof.prf) or in a user profile (user.prf).
- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the global server profile. For more information, see *Declaring Microsoft Access Connection Attributes From the Web Console* on page 18-3.

You can declare connections to more than one Microsoft Access database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to Microsoft Access Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Syntax **How to Declare Connection Attributes Manually**

Trusted authentication. The syntax is:

```
ENGINE [SQLMAC] SET CONNECTION_ATTRIBUTES datasource
```

Explicit authentication. For password protected datasources, the syntax is:

```
ENGINE [SQLMAC] SET CONNECTION_ATTRIBUTES datasource / , [password]
```

where:

SQLMAC

Indicates the Data Adapter for Microsoft Access. You can omit this value if you previously issued the SET SQLENGINE command.

data source

Is the name of the Microsoft Access data source you wish to access.

password

Is the password associated with the primary authorization ID.

Example **Declaring Connection Attributes**

The following SET CONNECTION_ATTRIBUTES command connects to the Microsoft Access database server named SAMPLESERVER with a password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile. The following SET CONNECTION_ATTRIBUTES command connects to the Microsoft Access database server named SAMPLESERVER using Password Passthru authentication:

```
ENGINE SQLMAC SET CONNECTION_ATTRIBUTES SAMPLESERVER /
```

Reference **Declaring Microsoft Access Connection Attributes From the Web Console**

Attribute	Description
Data source	Enter the data source name configured using the ODBC Driver Manager.
Password (Optional)	Enter a password only if the Microsoft Access data source is password protected.

Overriding the Default Connection

Once connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax How to Change the Default Connection

```
ENGINE [SQLMAC] SET DEFAULT_CONNECTION [connection]
```

where:

SQLMAC

Indicates the Data Adapter for Microsoft Access. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the service name defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following error message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following error message is issued: FOC1671, Command out of sequence.

Example Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the Microsoft Access database server named SAMPLENAME as the default Microsoft Access database server:

```
ENGINE SQLMAC SET DEFAULT_CONNECTION SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax How to Control the Connection Scope

```
ENGINE [SQLMAC] SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

SQLMAC

Indicates the Data Adapter for Microsoft Access. You can omit this value if you

previously issued the SET SQLENGINE command.

FIN

Disconnects automatically only after the session has been terminated. This value is the default.

COMMAND

Disconnects automatically after each request. Depending on how often the event occurs, the AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server; it is related to the operating system and the data source.

COMMIT

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Microsoft Access Metadata

When the server accesses a data source, it needs to know how to interpret the data that it finds. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Microsoft Access data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Microsoft Access table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Editing Metadata page opens.

To create a synonym, you must have configured the adapter. See *Configuring the Data Adapter for Microsoft Access* on page 18-2 for more information.

2. Expand the *Add* folder, expand the adapter folder, and then click a connection. The right pane displays table and view selection options:

Select All Tables/Views. Select this radio button to create synonyms for all tables and views. This value is the default.

Filter by Name, Owner and Table Type. Select this radio button to filter the tables or views for which to create synonyms.

Selecting this option adds the following:

Select all Tables/Views

Filter by Name, Owner and Table Type

Owner - Sample: abc%

Table name - Sample: abc%

Table type TABLE VIEW TABLE/VIEW

Select Tables

Owner. Enter a string for filtering the owners' IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owners' IDs begin with the letters ABC; %ABC to select tables or views whose owners' IDs end with the letters ABC; %ABC% to select tables or views whose owners' IDs contain the letters ABC at the beginning, middle, or end.

Table Name. Enter a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

Table Type. Select one of the following options: Tables (this is the default), Views, Tables/Views.

3. Click *Select Tables*. All tables that meet the specified criteria are displayed:

Prefix: SET CONVERSION LONGCHAR: TEXT (for Web Focus) ▼

Select Application Directory: ▼

All Select

Check	Default Synonym Name	Owner/Schema	Table Name	Type
<input type="checkbox"/>	<input type="text" value="NF29001"/>	EDAQA	NF29001	TABLE
<input type="checkbox"/>	<input type="text" value="NF29002"/>	EDAQA	NF29002	TABLE
<input type="checkbox"/>	<input type="text" value="NF29003"/>	EDAQA	NF29003	TABLE
<input type="checkbox"/>	<input type="text" value="NF29004"/>	EDAQA	NF29004	TABLE
<input type="checkbox"/>	<input type="text" value="NF29005"/>	EDAQA	NF29005	TABLE
<input type="checkbox"/>	<input type="text" value="NF29006"/>	EDAQA	NF29006	TABLE
<input type="checkbox"/>	<input type="text" value="NF29007"/>	EDAQA	NF29007	TABLE
<input type="checkbox"/>	<input type="text" value="NF29008"/>	EDAQA	NF29008	TABLE
<input type="checkbox"/>	<input type="text" value="NF29009"/>	EDAQA	NF29009	TABLE
<input type="checkbox"/>	<input type="text" value="NF29012"/>	EDAQA	NF29012	TABLE
<input type="checkbox"/>	<input type="text" value="NF29013"/>	EDAQA	NF29013	TABLE

4. If you have tables with identical table names, assign a prefix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave field blank.

5. From the SET CONVERSION LONGCHAR drop-down list, select: Text, Alpha, or BLOB. The default value is Text. See *Controlling the Mapping of Large Character Data Types* on page 18-13 for more information.
6. From the Select Application Directory drop-down list, select a directory. The default value is baseapp.
7. Complete your table or view selection:
To select all tables or views in the list, click **All**.
To select specific tables or views, click the corresponding check boxes.
8. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.

9. Click *Create Synonym*. Synonyms are created and added under the specified application directory.

Reference **Managing Synonyms**

In the navigation pane, click the name of the synonym to access the following options:

Sample Data	Retrieves up to 20 rows from the associated data source.
Properties	Displays a graphic representation of the synonym and enables you to edit its metadata.
Edit Master File	Enables you to manually edit the synonym's Master File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Master File, to update the synonym.
Edit Access File	Enables you to manually edit the synonym's Access File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Access File, to update the synonym.
Refresh	Regenerates the synonym. Use this option if structural changes were made to the data source.
Drop	Deletes the synonym.
Copy to	Copies the synonym to another application directory. Click the target directory from the resulting list.
Move to	Moves the synonym to another application directory. Click the target directory from the resulting list.

Syntax **How to Create a Synonym Manually**

```
CREATE SYNONYM app/synonym FOR table_view DBMS SQLMAC [AT
connection][NOCOLS]
END
```

where:

app

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, then this application name must not be used.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema).

SQLMAC

Indicates the Data Adapter for Microsoft Access.

AT connection

Is the service name as previously specified in a SET CONNECTION_ATTRIBUTES command. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

This parameter is optional. If specified, the CONNECTION attribute is added to the Access File.

NOCOLS

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.NF29004 DBMS SQLMAC AT MACDSN
END
```

Generated Master File nf29004.mas

```
FILE=NF29004 , SUFFIX=SQLMAC , $
SEGNAME=NF29004 , SEGTYPE=S0 , $
FIELD=DIVISION4 , DIVISION4 , I11 , I4 , MISSING=ON , $
FIELD=DIVISION_NA4 , DIVISION_NA4 , A25 , A25 , MISSING=ON , $
FIELD=DIVISION_HE4 , DIVISION_HE4 , I11 , I4 , MISSING=ON , $
```

Generated Access File nf29004.acx

```
SEGNAME=NF29004 , TABLENAME=NF29004 ,
CONNECTION=MACDSN , KEYS=1 , WRITE=YES , $
```

Reference Access File Keywords

Keyword	Description
TABLENAME	Identifies the Microsoft Access table name.
CONNECTION	<p>Indicates a previously declared connection. The syntax is:</p> <p><code>CONNECTION=connection</code></p> <p>CONNECTION=' ' indicates access to the local Microsoft Access database server.</p> <p>Absence of the CONNECTION attribute indicates access to the default database server.</p>

Data Type Support

The following chart provides information about the default mapping of Microsoft Access data types to server data types:

Microsoft Access Data Types	Data Type		Remarks
	Usage	Actual	
TEXT (<i>n</i>) in (1...255)	A255	A255 H	
MEMO (<i>n</i>) in (1...65,535)	Am TX50	Am TX	$m=2*n$.
AUTONUMBER (Long Integer)	I6	I4	
YES/NO	I11	I4	
OLE OBJECT (drawings, waveform audio files, bitmapped graphics)	BLOB	BLOB	Supported via API
HYPERLINK	Am TX50	Am TX	$m=2*n$.
CURRENCY (range of -922337203685477.5808 to +922337203685477.5808)			
General	P21.4	P10	
Currency	P21.4	P10	
Euro	P21.4	P10	
Fixed	P21.4	P10	
Standard	P21.4	P10	
Percent	P21.4	P10	
Scientific	P21.4	P10	
NUMBER (Subtypes)			
Byte (range 0 to 255)	I6	I4	
Integer (range of 32,768 to 32,767)	I6	I4	

Microsoft Access Data Types	Data Type		Remarks
	Usage	Actual	
Long Integer (range of 2,147,483,648 to 2,147,483,647)	I11	I4	
Single (range of 3.4×10^{38} to $+3.4 \times 10^{38}$)	D20.2	D8	
Double (range of 1.797×10^{308} to $+1.797 \times 10^{308}$)	D20.2	D8	
Decimal (-10^{28} to $+10^{28-1}$)	P19	P10	
Replication ID	A38	A38	
DATE/TIME			
General Date	HYYMDS	HYYMDS	
Long Date	HYYMDS	HYYMDS	
Medium Date	HYYMDS	HYYMDS	
Short Date	HYYMDS	HYYMDS	
Long Time	HYYMDS	HYYMDS	
Medium Time	HYYMDS	HYYMDS	
Short Time	HYYMDS	HYYMDS	

Controlling the Mapping of Large Character Data Types

The SET parameter CONVERSION LONGCHAR controls the mapping of supported Microsoft Access data types listed below. By default, the server maps these data types as alphanumeric (A). The server data type A supports a maximum of 4096 characters for TABLE/MODIFY and 32768 characters for API applications.

The following table lists data type mappings based on the value of LONGCHAR:

Microsoft Access Data Type	Remarks	LONGCHAR ALPHA or BLOB		LONGCHAR TEXT	
		Usage	Actual	Usage	Actual
CHAR (<i>n</i>)	<i>n</i> is an integer between 1 and 2000	<i>An</i>	<i>An</i>	TX50	TX
VARCHAR (<i>n</i>)	<i>n</i> is an integer between 1 and 4000	<i>An</i>	<i>An</i>	TX50	TX
VARCHAR2 (<i>n</i>)	<i>n</i> is an integer between 1 and 4000	<i>An</i>	<i>An</i>	TX50	TX
RAW (<i>n</i>)	<i>n</i> is an integer between 1 and 2000 $m = 2 * n$	<i>Am</i>	<i>Am</i>	TX50	TX

Syntax

How to Control the Mapping of Large Character Data Types

```
ENGINE [SQLMAC] SET CONVERSION LONGCHAR {ALPHA|TEXT|BLOB}
```

where:

SQLMAC

Indicates the Data Adapter for Microsoft Access. You can omit this value if you previously issued the SET SQLENGINE command.

ALPHA

Maps the Microsoft Access data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as alphanumeric (A). This value is the default.

TEXT

Maps the Microsoft Access data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as text (TX). Use this value for WebFOCUS applications.

BLOB

For UNIX, Windows, OpenVMS, and OS/400, is identical to ALPHA. That is, it maps the Microsoft Access data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as alphanumeric (A).

For OS/390 and z/OS, maps the Microsoft Access data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as binary large object (BLOB).

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Syntax

How to Override Default Precision and Scale

```
ENGINE [SQLMAC] SET CONVERSION RESET  
ENGINE [SQLMAC] SET CONVERSION format RESET  
ENGINE [SQLMAC] SET CONVERSION format [PRECISION pp [ss]]  
ENGINE [SQLMAC] SET CONVERSION format [PRECISION MAX]
```

where:

SQLMAC

Indicates the Data Adapter for Microsoft Access. You can omit this value if you previously issued the SET SQLENGINE command.

RESET

Returns any previously specified precision and scale values to the data adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

format

Is any valid format supported by the data source. Possible values are:

INTEGER which indicates that the command applies only to INT columns.

DECIMAL which indicates that the command applies only to DECIMAL columns.

FLOAT which indicates that the command applies only to double precision floating point columns.

pp

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

ss

Is the scale. This is valid with DECIMAL and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. If the scale is not required, you must set mm to 0 (zero).

MAX

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example **Setting the Precision and Scale Attributes**

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE SQLMAC SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLMAC SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE SQLMAC SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLMAC SET CONVERSION RESET
```

Customizing the Microsoft Access Environment

The Data Adapter for Microsoft Access provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Syntax **How to Obtain the Number of Rows Updated or Deleted**

```
ENGINE [SQLMAC] SET PASSRECS {ON|OFF}
```

where:

SQLMAC

Indicates the Data Adapter for Microsoft Access. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. This value is the default.

OFF

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

CHAPTER 19

Getting Started in Microsoft SQL Server

Topics:

- Preparing the Microsoft SQL Server Environment
- Configuring the Data Adapter for Microsoft SQL Server
- Managing Microsoft SQL Server Metadata
- Customizing the Microsoft SQL Server Environment
- Calling a Microsoft SQL Server Stored Procedure Using SQL Passthru
- Microsoft SQL Server Compatibility With ODBC

The Data Adapter for Microsoft SQL Server allows applications to access Microsoft SQL Server data sources. The adapter converts data or application requests into native Microsoft SQL Server statements and returns optimized answer sets to the requesting program.

Preparing the Microsoft SQL Server Environment

The Microsoft SQL Server environment is set up during the installation of the Microsoft SQL Server, Client, and MDAC software.

Accessing Microsoft SQL Server Remotely

You can access Microsoft SQL Server on a remote node. To access Microsoft SQL Server remotely, you must:

- Locally install MDAC 2.6 or higher.
- Know the name of the remote Microsoft SQL Server instance.

All Microsoft SQL Servers installed are defined by using a unique NetBEUI name. The server can access any Microsoft SQL Server on the network, provided you define a valid user ID and password, as well as the name of the Microsoft SQL Server instance. You can define these parameters in either the server's global profile or a user profile.

XA Support

Read/write applications accessing Microsoft SQL Server data sources are able to perform transactions managed in XA-compliant mode.

To activate the XA Transaction Management feature, the server has to be configured in Transaction Coordination Mode, using the Web console configuration functions. Using Transaction Coordination Mode guarantees the integrity of data modification on all of the involved DBMSs and protects part of the data modifications from being committed on one DBMS and terminated on another.

For complete documentation on XA compliance, see the XA Support appendix.

Configuring the Data Adapter for Microsoft SQL Server

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to an Microsoft SQL Server database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Manually add the command in the global server profile (edasprof.prf) or in a user profile (user.prf).
- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the global server profile. For more information, see *Declaring Microsoft SQL Server Connection Attributes From the Web Console* on page 19-6.

You can declare connections to more than one Microsoft SQL Server database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to Microsoft SQL Server Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Syntax How to Declare Connection Attributes Manually

Explicit authentication. The user ID and password are explicitly specified for each connection and passed to Microsoft SQL Server, at connection time, for authentication. The syntax is

```
ENGINE [SQLMSS] SET CONNECTION_ATTRIBUTES [connection]
[server]/user_ID,password [;dbname][:provider_string]
```

Password passthru authentication. The user ID and password are explicitly specified for each connection and passed to Microsoft SQL Server, at connection time, for authentication. This option requires that the server be started with security off. The syntax is

```
ENGINE [SQLMSS] SET CONNECTION_ATTRIBUTES [connection]
[server]/[:dbname][:provider_string]
```

Trusted authentication. The adapter connects to Microsoft SQL Server as a Windows login using the credentials of the Windows user impersonated by the server data access agent. The syntax is

```
ENGINE [SQLMSS] SET CONNECTION_ATTRIBUTES [connection]  
[server] / [, [dbname] [:provider_string]
```

where:

SQLMSS

Indicates the Data Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is a logical name (or a data source name) used to identify this particular set of attributes.

If you plan to have only one connection to Microsoft SQL Server, this parameter is optional. If not specified, the local database server serves as the default connection.

server

Is the name of the machine where Microsoft SQL Server is running. If that machine has more than one instance of Microsoft SQL Server installed, provide the server name and instance as follows: *server*\instance.

When specifying the server name and the instance name, we recommend that you enclose the value in single quotation marks (').

user_ID

Is the primary authorization ID by which you are known to Microsoft SQL Server.

password

Is the password associated with the primary authorization ID.

dbname

Also referred to as *schema*, is the name of the Microsoft SQL Server database used for this connection. The database name, including path, must be enclosed in single quotes.

provider_string

Is the Microsoft SQL Server provider string used to specify additional connection option, such as the name of the network library. Note that this parameter must be preceded by a colon and enclosed in single quotation marks. This parameter is optional.

Note: Enclose values that contain special characters in single quotation marks ('). If a value contains a single quotation mark, this quotation mark must be preceded by another single quotation mark, resulting in two single quotation marks in succession. For example, to specify the user ID Mary O'Brien, which contains both a blank and a single quotation mark, enter: 'Mary O' Brian'

Example **Declaring Connection Attributes**

The following SET CONNECTION_ATTRIBUTES command allow the application to access the Microsoft SQL Server database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLMSS SET CONNECTION_ATTRIBUTES SAMPLESERVER/MYUSER,PASS
```

The following SET CONNECTION_ATTRIBUTES command connects to the Microsoft SQL Server database server named SAMPLESERVER using Password Passthru authentication:

```
ENGINE SQLMSS SET CONNECTION_ATTRIBUTES SAMPLESERVER/
```

The following SET CONNECTION_ATTRIBUTES command connects to a local Microsoft SQL Server database server using operating system authentication:

```
ENGINE SQLMSS SET CONNECTION_ATTRIBUTES /,
```

Reference Declaring Microsoft SQL Server Connection Attributes From the Web Console

Attribute	Description
Connection name	Is a logical name used to identify this particular set of connection attributes.
Server	Is the name of the machine where Microsoft SQL Server is running. If that machine has more than one instance of Microsoft SQL Server installed, provide the server name and the instance name as follows: server\instance.
Security	<p>There are three methods by which a user can be authenticated when connecting to an Microsoft SQL Server:</p> <p>Explicit. The user ID and password are explicitly specified for each connection and passed to Microsoft SQL Server, at connection time, for authentication as a standard login.</p> <p>This option requires that SQL Server security be set to: SQL Server and Windows.</p> <p>Password Passthru. The user ID and password received from the client application are passed to Microsoft SQL Server, at connection time, for authentication as a standard login.</p> <p>This option requires that the server be started with security off and that SQL Server security be set to: SQL Server and Windows.</p> <p>Trusted. The adapter connects to Microsoft SQL Server as a Windows login using the credentials of the Windows user impersonated by the server data access agent.</p> <p>This option works with either of the SQL Server security settings.</p>
User	Is the authorization ID by which the user is known at the instance of Microsoft SQL Server.
Password	Is the password associated with the authorization ID. The password is stored in encrypted form.
Dbname	<p>Is the name of the default database for the connection. This value is used when a data object is not qualified with the database name.</p> <p>This parameter is optional. If not specified, it defaults to the database associated with the authorization ID.</p>

Overriding the Default Connection

Once connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax How to Change the Default Connection

```
ENGINE [SQLMSS] SET DEFAULT_CONNECTION [connection]
```

where:

SQLMSS

Indicates the Data Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the service name defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following error message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following error message is issued: FOC1671, Command out of sequence.

Example Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the Microsoft SQL Server database server named SAMPLENAME as the default Microsoft SQL Server database server:

```
ENGINE SQLMSS SET DEFAULT_CONNECTION SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax **How to Control the Connection Scope**

```
ENGINE [SQLMSS] SET AUTODISCONNECT ON {FIN|COMMAND}
```

where:

SQLMSS

Indicates the Data Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

FIN

Disconnects automatically only after the session has been terminated. This value is the default.

COMMAND

Disconnects automatically after each request. Depending on how often the event occurs, the AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server; it is related to the operating system and the data source.

Managing Microsoft SQL Server Metadata

When the server accesses a data source, it needs to know how to interpret the data that it finds. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Microsoft SQL Server data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Microsoft SQL Server table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click Metadata. The Editing Metadata page opens.

To create a synonym, you must have configured the adapter. See *Configuring the Data Adapter for Microsoft SQL Server* on page 19-3 for more information.

2. Expand the *Add* folder, expand the adapter folder, and then click a connection. The right pane displays table and view selection options:

Select Database. A drop-down list shows all databases available on the selected Microsoft SQL Server. Select the database that contains the tables or views for which you want to create a synonym.

Select All Tables/Views. Select this radio button to create synonyms for all tables and views. This value is the default.

Filter by Name, Owner and Table Type. Select this radio button to filter the tables or views for which to create synonyms.

Selecting this option adds the following:

The screenshot shows a blue-themed web console interface for selecting tables and views. At the top, there is a 'Select Database:' label followed by a dropdown menu containing 'QAEDA'. Below this are two radio buttons: 'Select all Tables/Views' (which is unselected) and 'Filter by Name, Owner and Table Type' (which is selected). Under the selected radio button, there are three input fields: 'Owner' with a text box and a hint '- Sample: abc%', 'Table Name' with a text box containing 'NF%' and a hint '- Sample: abc%', and 'Table type' with three radio buttons: 'TABLE' (selected), 'VIEW' (unselected), and 'TABLE/VIEW' (unselected). At the bottom of the form is a button labeled 'Select Tables'.

Owner. Enter a string for filtering the owners' IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owners' IDs begin with the letters ABC; %ABC to select tables or views whose owners' IDs end with the letters ABC; %ABC% to select tables or views whose owners' IDs contain the letters ABC at the beginning, middle, or end.

Table Name. Enter a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

Table Type. Select one of the following options: Tables (this is the default), Views, Tables/Views.

3. Click *Select Tables*. All tables that meet the specified criteria are displayed:

Check	Default Synonym Name	Owner/Schema	Table Name	Type
<input type="checkbox"/>	NF29001	EDAQA	NF29001	TABLE
<input type="checkbox"/>	NF29002	EDAQA	NF29002	TABLE
<input type="checkbox"/>	NF29003	EDAQA	NF29003	TABLE
<input type="checkbox"/>	NF29004	EDAQA	NF29004	TABLE
<input type="checkbox"/>	NF29005	EDAQA	NF29005	TABLE
<input type="checkbox"/>	NF29006	EDAQA	NF29006	TABLE
<input type="checkbox"/>	NF29007	EDAQA	NF29007	TABLE
<input type="checkbox"/>	NF29008	EDAQA	NF29008	TABLE
<input type="checkbox"/>	NF29009	EDAQA	NF29009	TABLE
<input type="checkbox"/>	NF29012	EDAQA	NF29012	TABLE
<input type="checkbox"/>	NF29013	EDAQA	NF29013	TABLE

4. If you have tables with identical table names, assign a prefix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave field blank.

5. From the SET CONVERSION LONGCHAR drop-down list, select: Text, Alpha, or BLOB. The default value is Text. See *Controlling the Mapping of Large Character Data Types* on page 19-16 for more information.

6. From the Select Application Directory drop-down list, select a directory. The default value is baseapp.
7. Complete your table or view selection:
 - To select all tables or views in the list, click **All**.
 - To select specific tables or views, click the corresponding check boxes.
8. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.
9. Click *Create Synonym*. Synonyms are created and added under the specified application directory.

Reference **Managing Synonyms**

In the navigation pane, click the name of the synonym to access the following options:

Sample Data	Retrieves up to 20 rows from the associated data source.
Properties	Displays a graphic representation of the synonym and enables you to edit its metadata.
Edit Master File	Enables you to manually edit the synonym's Master File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Master File, to update the synonym.
Edit Access File	Enables you to manually edit the synonym's Access File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Access File, to update the synonym.
Refresh	Regenerates the synonym. Use this option if structural changes were made to the data source.
Drop	Deletes the synonym.
Copy to	Copies the synonym to another application directory. Click the target directory from the resulting list.
Move to	Moves the synonym to another application directory. Click the target directory from the resulting list.

Syntax **How to Create a Synonym Manually**

```
CREATE SYNONYM app/synonym FOR table_view DBMS SQLMSS [AT connection]  
END
```

where:

app

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, then this application name must not be used.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema) as follows:

```
[[database.]owner.]table
```

SQLMSS

Indicates the Data Adapter for Microsoft SQL Server.

AT *connection*

Is the service name as previously specified in a SET CONNECTION_ATTRIBUTES command. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

This parameter is optional. If specified, the CONNECTION attribute is added to the Access File.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example Using CREATE SYNONYM

```
CREATE SYNONYM baseapp/nf29004 FOR edaga.nf29004 DBMS SQLMSS AT connmss
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION ,SUFFIX=SQLMSS,$
SEGNAME=SEG1_4 ,SEGTYPE=S0 ,$
FIELD=DIVISION4 ,DIVISION4 ,I9 ,I4 ,MISSING=OFF ,$
FIELD=DIVISION_NA4,DIVISION_NA4 ,A25 ,A25 ,MISSING=ON ,$
FIELD=DIVISION_HE4,DIVISION_HE4 ,I9 ,I4 ,MISSING=ON , $
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4, TABLENAME=edaga.nf29004,
CONNECTION=connmss, KEYS=1, WRITE=YES,$
```

Reference Access File Keywords

Attribute	Description
TABLENAME	Identifies the Microsoft SQL Server table. The table name can be fully qualified as follows: TABLENAME=[[database.]owner.]table
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=connection

Data Type Support

The following table lists how the server maps Microsoft SQL Server data types. Note that you can:

- Control the mapping of large character data types.
- Control the mapping of variable-length data types.
- Enable National Language Support.
- Change the precision and scale of numeric columns.

Be sure to review those options carefully as they affect the mapping of data types.

Microsoft SQL Server Data Type	Data Type		Remarks
	Usage	Actual	
CHAR (<i>n</i>)	<i>An</i>	<i>An</i>	<i>n</i> is an integer between 1 and 8000
NCHAR (<i>n</i>)	<i>An</i>	<i>An</i>	<i>n</i> is an integer between 1 and 4000
VARCHAR (<i>n</i>)	<i>An</i>	<i>An</i>	<i>n</i> is an integer between 1 and 8000
NVARCHAR (<i>n</i>)	<i>An</i>	<i>An</i>	<i>n</i> is an integer between 1 and 4000
TEXT	A32767	A32767	
NTEXT	TX50	TX	
BINARY(<i>n</i>)	<i>Am</i>	<i>Am</i>	<i>n</i> is an integer between 1 and 8000 <i>m</i> = 2 * <i>n</i>
VARBINARY(<i>n</i>)	<i>Am</i>	<i>Am</i>	<i>n</i> is an integer between 1 and 8000 <i>m</i> = 2 * <i>n</i>
SQL_VARIANT	A8000	A8000	
UNIQUEIDENTIFIER (GUID)	A38	A38	
TIMESTAMP	A16	A16	Supported as Read-only
DATETIME	HYYMDs	HYYMDs	range: 1/1/1753 to 12/31/9999
SMALLDATETIME	HYYMDI	HYYMDI	range: 1/1/1900 thru 6/6/2079
INT	I11	I4	range: -231 to 231-1
BIGINT	P20	P10	range: 263 to 263-1

Microsoft SQL Server Data Type	Data Type		Remarks
	Usage	Actual	
SMALLINT	I6	I4	range: -215 to 215-1
TINYINT	I6	I4	range: 0 to 255
BIT	I11	I4	-1 for "True" and 0 for "False"
DECIMAL (p, s)	Pn.m	Pk	<p>p is an integer between 1 and 38 s is an integer between 0 and p</p> <p>If s is 0 and p is between 1 and 31, $n = p + 1$ If s is 0 and p is between 32 and 38, $n = 32$</p> <p>If s is greater than 0 and p is between 1 and 31, $n = p + 2$ and $m = s$ If s is greater than 0 and p is between 32 and 38, $n = 33$ and $m = 31$</p> <p>If p is between 1 and 31, $k = (p / 2) + 1$ If p is between 32 and 38, $k = 16$</p> <p>Note: If the column is nullable, p is greater than or equal to 8</p>
NUMERIC (p, s)	Pn.m	Pk	<p>p is an integer between 1 and 38 s is an integer between 0 and p</p> <p>If s is 0 and p is between 1 and 31, $n = p + 1$ If s is 0 and p is between 32 and 38, $n = 32$</p> <p>If s is greater than 0 and p is between 1 and 31, $n = p + 2$ and $m = s$ If s is greater than 0 and p is between 32 and 38, $n = 33$ and $m = 31$</p> <p>If p is between 1 and 31, $k = (p / 2) + 1$ If p is between 32 and 38, $k = 16$</p> <p>Note: If the column is nullable, p is greater than or equal to 8</p>
MONEY	P21.4	P10	range: -263 to 263-1
SMALLMONEY	P12.4	P8	range: -214,748.3648 to 214,748.3647

Microsoft SQL Server Data Type	Data Type		Remarks
	Usage	Actual	
FLOAT	D20.2	D8	range: -1.79E+308 to 1.79E+308
REAL	D20.2	D8	range: -3.40E+38 to 3.40E+38
IMAGE	BLOB	BLOB	length: 231-1 Supported through iWay API

Controlling the Mapping of Large Character Data Types

The SET parameter CONVERSION LONGCHAR controls the mapping of supported Microsoft SQL Server data types listed below. By default, the server maps these data types as alphanumeric (A). The server data type A supports a maximum of 4096 characters for TABLE/MODIFY and 32768 characters for API applications.

The following table lists data type mappings based on the value of LONGCHAR:

Microsoft SQL Server Data Type	Remarks	LONGCHAR ALPHA or BLOB		LONGCHAR TEXT	
		Usage	Actual	Usage	Actual
CHAR (<i>n</i>)	<i>n</i> is an integer between 1 and 256	<i>An</i>	<i>An</i>	<i>An</i>	<i>An</i>
	<i>n</i> is an integer between 257 and 8000	<i>An</i>	<i>An</i>	TX50	TX
NCHAR (<i>n</i>)	<i>n</i> is an integer between 1 and 128	<i>An</i>	<i>An</i>	<i>An</i>	<i>An</i>
	<i>n</i> is an integer between 129 and 4000	<i>An</i>	<i>An</i>	TX50	TX
VARCHAR (<i>n</i>)	<i>n</i> is an integer between 1 and 256	<i>An</i>	<i>An</i>	<i>An</i>	<i>An</i>
	<i>n</i> is an integer between 257 and 8000	<i>An</i>	<i>An</i>	TX50	TX

Microsoft SQL Server Data Type	Remarks	LONGCHAR ALPHA or BLOB		LONGCHAR TEXT	
		Usage	Actual	Usage	Actual
NVARCHAR (<i>n</i>)	<i>n</i> is an integer between 1 and 128	<i>An</i>	<i>An</i>	<i>An</i>	<i>An</i>
	<i>n</i> is an integer between 129 and 4000	<i>An</i>	<i>An</i>	TX50	TX
TEXT		A32767	A32767	TX50	TX
BINARY (<i>n</i>)	<i>n</i> is an integer between 1 and 8000 $m = 2 * n$	<i>Am</i>	<i>Am</i>	TX50	TX
VARBINARY (<i>n</i>)	<i>n</i> is an integer between 1 and 8000 $m = 2 * n$	<i>Am</i>	<i>Am</i>	TX50	TX
SQL_VARIANT		A8000	A8000	TX50	TX

Syntax**How to Control the Mapping of Large Character Data Types**

ENGINE [SQLMSS] SET CONVERSION LONGCHAR {[ALPHA](#)|[TEXT](#)|[BLOB](#)}

where:

SQLMSS

Indicates the Data Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

ALPHA

Maps the Microsoft SQL Server data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as alphanumeric (A). This value is the default.

TEXT

Maps the Microsoft SQL Server data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as text (TX). Use this value for WebFOCUS applications.

BLOB

For Windows is identical to ALPHA. That is, it maps the Microsoft SQL Server data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as alphanumeric (A).

Controlling the Mapping of Variable-Length Data Types

The SET parameter VARCHAR controls the mapping of the Microsoft SQL Server data types VARCHAR, VARCHAR2, and NVARCHAR2. By default, the server maps these data types as alphanumeric (A).

The following table lists data type mappings based on the value of VARCHAR:

Microsoft SQL Server Data Type	Remarks	VARCHAR ON		VARCHAR OFF	
		Usage	Actual	Usage	Actual
VARCHAR (<i>n</i>)	<i>n</i> is an integer between 1 and 8000	<i>AnV</i>	<i>AnV</i>	<i>An</i>	<i>An</i>
NVARCHAR (<i>n</i>)	<i>n</i> is an integer between 1 and 4000	<i>AnV</i>	<i>AnV</i>	<i>An</i>	<i>An</i>
VARBINARY (<i>n</i>)	<i>n</i> is an integer between 1 and 8000 $m = 2 * n$	<i>AmV</i>	<i>AmV</i>	<i>Am</i>	<i>Am</i>

Syntax

How to Control the Mapping of Variable-Length Data Types

```
ENGINE [SQLMSS] SET VARCHAR {ON|OFF}
```

where:

SQLMSS

Indicates the Data Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Maps the Microsoft SQL Server data types VARCHAR, VARCHAR2, and NVARCHAR2 as variable-length alphanumeric (*AnV*).

OFF

Maps the Microsoft SQL Server data types VARCHAR, VARCHAR2, and NVARCHAR2 as alphanumeric (A). This value is the default.

Enabling National Language Support

The SET parameter NCHAR indicates whether the character set is single-byte, double-byte, or triple-byte. The NCHAR setting affects the mapping of NCHAR and NVARCHAR data types.

The following chart lists data type mappings based on the value of NCHAR.

Microsoft SQL Server Data Type	Remarks	NCHAR SBCS		NCHAR DBCS		NCHAR TBCS	
		Usage	Actual	Usage	Actual	Usage	Actual
NCHAR (<i>n</i>)	<i>n</i> is an integer between 1 and 4000 $d = 2 * n$ $t = 3 * n$	<i>An</i>	<i>An</i>	<i>Ad</i>	<i>Ad</i>	<i>At</i>	<i>At</i>
NVARCHAR (<i>n</i>)	<i>n</i> is an integer between 1 and 4000 $d = 2 * n$ $t = 3 * n$	<i>An</i>	<i>An</i>	<i>Ad</i>	<i>Ad</i>	<i>At</i>	<i>At</i>

Syntax How to Enable National Language Support

```
ENGINE [SQLMSS] SET NCHAR {SBCS|DBCS|TBCS}
```

where:

SQLMSS

Indicates the Data Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

SBCS

Indicates a single-byte character set. This value is the default.

DBCS

Indicates a double-byte character set.

TBCS

Indicates a triple-byte character set.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Syntax

How to Override Default Precision and Scale

```
ENGINE [SQLMSS] SET CONVERSION RESET  
ENGINE [SQLMSS] SET CONVERSION format RESET  
ENGINE [SQLMSS] SET CONVERSION format [PRECISION pp [ss]]  
ENGINE [SQLMSS] SET CONVERSION format [PRECISION MAX]
```

where:

SQLMSS

Indicates the Data Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

RESET

Returns any previously specified precision and scale values to the data adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

format

Is any valid format supported by the data source. Possible values are:

INTEGER which indicates that the command applies only to INT columns.

DECIMAL which indicates that the command applies only to DECIMAL columns.

REAL which indicates that the command applies only to single precision floating point columns.

FLOAT which indicates that the command applies only to double precision floating point columns.

pp

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

ss

Is the scale. This is valid with DECIMAL and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. If the scale is not required, you must set mm to 0 (zero).

MAX

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
REAL	9
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example **Setting the Precision and Scale Attributes**

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE SQLMSS SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLMSS SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE SQLMSS SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLMSS SET CONVERSION RESET
```

Support of Read-Only Fields

CREATE SYNONYM creates a field description with FIELDTYPE=R for Microsoft SQL Server columns created as TIMESTAMP or columns with the IDENTITY attribute. These fields are read-only. When executing a MAINTAIN or MODIFY procedure, the adapter suppresses all write operations against columns marked in the Master File with FIELDTYPE=R.

Example Supporting a Read-Only Field

This example creates a table in which the first column has the IDENTITY property and the second column is a timestamp column:

```
CREATE TABLE TAB1  
  (idproptab int IDENTITY (1,1), timstmp timestamp)
```

CREATE SYNONYM generates the following Master File for this table:

```
FILE=TAB1 , SUFFIX=SQLMSS , $  
SEGNAME=TAB1 , SEGTYPE=S0 , $  
FIELD=IDPROPTAB , idproptab, I11,I4 ,MISSING=OFF, FIELDTYPE=R, $  
FIELD=TIMSTMP , timstmp, A16,A16,MISSING=ON , FIELDTYPE=R, $
```

Customizing the Microsoft SQL Server Environment

The Data Adapter for Microsoft SQL Server provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to Microsoft SQL Server.

Syntax **How to Issue the TIMEOUT Command**

The syntax is

```
ENGINE [SQLMSS] SET COMMANDTIMEOUT {nn|0}
```

where:

SQLMSS

Indicates the SQLENGINE setting. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a timeout occurs. The default value is 30.

0

Represents an infinite period to wait for response.

Note: If you do not specify a COMMANDTIMEOUT value, you default to the current Microsoft SQL Server default setting.

Specifying the Cursor Type

You can use the SET CURSORS command to specify the type of cursors for retrieval.

Syntax **How to Specify the Cursor Type**

```
ENGINE [SQLMSS] SET CURSORS [CLIENT|SERVER]
```

where:

CLIENT

Uses Microsoft SQL Server client-side cursors for retrieving data. Client-side cursors normally demonstrate the best performance for data retrieval and benefit the Microsoft SQL Server process. However, except in TRANSACTIONS AUTOCOMMITTED mode, using client-side cursors prevents a server agent from simultaneously reading more than one answer set from the same instance of Microsoft SQL Server.

SERVER

Uses Microsoft SQL Server server-side cursors for retrieving data. Server-side cursors demonstrate lower performance than client cursors. However, setting a high FETCHSIZE factor (100 is the adapter default) improves their performance dramatically making them almost as fast as client-side cursors. Client-side cursors are recommended wherever possible in order to take the load off the Microsoft SQL Server process.

blank

Uses client-side cursors in TRANSACTIONS AUTOCOMMITTED mode and server-side cursors otherwise. This value is the default.

Specifying the Block Size for Array Retrieval

The Data Adapter for Microsoft SQL Server supports array retrieval from result sets produced by executing SELECT queries or stored procedures. This technique substantially reduces network traffic and CPU utilization.

Using high values increases the efficiency of requests involving many rows, at the cost of higher virtual storage requirements. A value higher than 100 is not recommended because the increased efficiency it would provide is generally negligible.

Syntax How to Specify the Block Size for Array Retrieval

The block size for a SELECT request applies to TABLE FILE requests, MODIFY requests, MATCH requests, and DIRECT SQL SELECT statements.

```
ENGINE [SQLMSS] SET FETCHSIZE n
```

where:

SQLMSS

Indicates the Data Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is the number of rows to be retrieved at once using array retrieval techniques. Accepted values are 1 to 5000. The default is 20. If the result set contains a column that has to be processed as a CLOB or a BLOB, the fetchsize value used for that result set is 1.

Syntax How to Specify the Block Size for Inserting Rows

In combination with LOADONLY, the block size for an INSERT applies to MODIFY INCLUDE requests. INSERTSIZE is also supported for parameterized DIRECT SQL INSERT statements.

```
ENGINE [SQLMSS] SET INSERTSIZE n
```

where:

SQLMSS

Indicates the Data Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is the number of rows to be inserted using array insert techniques. Accepted values are 1 to 5000. The default is 1. If the result set contains a column that has to be processed as a BLOB, the insertsize value used for that result set is 1.

Specifying the Login Wait Time

You can use the LOGINTIMEOUT command to specify the number of seconds the adapter will wait for a response from Microsoft SQL Server at connect time. **Note:** For compatibility with previous releases of the adapter, TIMEOUT is available as a synonym for LOGINTIMEOUT.

Syntax How to Specify the Login Wait Time

```
ENGINE [SQLMSS] SET LOGINTIMEOUT|TIMEOUT {nn|0}
```

where:

SQLMSS

Indicates the Data Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a timeout occurs. The default value is approximately 15 seconds.

0

Represents an infinite period to wait for login response.

Activating NONBLOCK Mode

The Data Adapter for Microsoft SQL Server has the ability to issue calls in NONBLOCK mode. The default behavior is BLOCK mode.

This feature allows the adapter to react to a client request to cancel a query while the adapter is waiting on engine processing. This wait state usually occurs during SQL parsing, before the first row of an answer set is ready for delivery to the adapter or while waiting for access to an object that has been locked by another application.

Syntax **How to Activate NONBLOCK Mode**

```
ENGINE [SQLMSS] SET NONBLOCK {0|n}
```

where:

SQLMSS

Indicates the Data Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is a positive numeric number. A value of 0, the default, means that the adapter will operate in BLOCK mode. A value of 1 or greater activates the NONBLOCK calling and specifies the time, in seconds, that the adapter will wait between each time it checks to see if the:

- Query has been executed.
- Client application has requested the cancellation of a query.
- Kill Session button on the Web Console is pressed.

Note: A value of 1 or 2 should be sufficient for normal operations.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Syntax **How to Obtain the Number of Rows Updated or Deleted**

```
ENGINE [SQLMSS] SET PASSRECS {ON|OFF}
```

where:

[SQLMSS](#)

Indicates the Data Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

[ON](#)

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. This value is the default.

[OFF](#)

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Controlling Transactions

You can use the SET TRANSACTIONS command to control how the adapter handles transactions.

Syntax **How to Control Transactions**

```
ENGINE [SQLMSS] SET TRANSACTIONS {LOCAL|DISTRIBUTED|AUTOCOMMITTED}
```

where:

SQLMSS

Indicates the Data Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

LOCAL

Indicates that the adapter implicitly starts a local transaction on each of the connections where any work is performed. At the time of COMMIT or ROLLBACK, or at the end of the server session, the adapter commits or aborts the work on each connection consecutively.

DISTRIBUTED

Indicates that the adapter implicitly invokes Microsoft Distributed Transactions Coordinator (DTC) to create a single distributed transaction within which to perform all work on all the connections. At the time of COMMIT or ROLLBACK, or at the end of the server session, the adapter invokes DTC to execute the two-phase commit or rollback protocol. For this purpose, the DTC service must be started on the machine where the server is running and also on all the machines where involved instances of Microsoft SQL Server reside.

This mode is recommended for read-write applications that perform updates on multiple connections simultaneously.

AUTOCOMMITTED

Indicates that each individual operation with Microsoft SQL Server is immediately committed (if successful) or rolled back (in case of errors) by the SQL Server. This is recommended for read-only applications for performance considerations. It is not recommended for read-write applications because in this mode it is impossible to roll back a logical unit of work that consists of several operations.

Calling a Microsoft SQL Server Stored Procedure Using SQL Passthru

Microsoft SQL Server stored procedures are supported via SQL Passthru. These procedures need to be developed within Microsoft SQL Server using the CREATE PROCEDURE command.

Example Calling a Stored Procedure

The supported syntax to call a stored procedure is shown below.

```
ENGINE SQLMSS
EX SAMPLE PARM1 , PARM2 , PARM3 . . . ;
TABLE ON TABLE PCHOLD
END
```

The server supports invocation of stored procedures written according to the following rules:

- Only scalar input parameters are allowed, but not required.
- Output parameters are not allowed, except for optional cursor type parameters.
- A cursor used for output parameters must be defined with:
 - The TYPE statement in a PROCEDURE.
 - An associated record layout of the answer set to be returned.
- The cursor must be opened in the procedure.
- No fetching is allowed from the output parameter cursors in the stored procedure. The Data Adapter for Microsoft SQL Server fetches the answer set.
- Any error messages must be issued using the RAISE APPLICATION ERROR method.

Microsoft SQL Server Compatibility With ODBC

Release 5.2 of the Data Adapter for Microsoft SQL Server is based on OLE DB, the Microsoft-recommended API for developing high-performance components. The adapter uses the OLE DB Provider for SQL Server (SQLOLEDB), a native, high-performance provider that directly accesses the SQL Server TDS protocol.

Earlier releases of the adapter were based on the ODBC API. Release 5.2 of the adapter supports all the functionality of the earlier, ODBC-based versions. In addition, it introduces the following enhancements:

- Support for distributed transactions.
- Array Retrieval.
- Command execution timeout.
- Fast load. This feature, enabled automatically, improves the performance of load-only ETL applications and MODIFY procedures.
- XA transaction support. For more information, see the Data Adapter Administration manual.

Differences between adapter functionality under OLE DB and ODBC exist in the following areas:

- Connection attributes.
- Cursors.
- Mapping of UNIQUEIDENTIFIER and BIT data types.

If you are using ODBC, be sure to review these topics.

Microsoft SQL Server Connection Attributes With ODBC

An ODBC data source declaration contains the following information: the instance of Microsoft SQL Server, the default database, the network libraries, and so on. In earlier, ODBC-based releases of the adapter, the SET CONNECTION_ATTRIBUTES command required only the name of an ODBC data source (User or System DNS) and authentication parameters.

Unlike ODBC data sources, OLE DB data sources directly reference instances of Microsoft SQL Server. To connect to an OLE DB data source, the adapter has to specify all the pertinent parameters at connection time. For this reason, the syntax of the SET CONNECTION_ATTRIBUTES command has been expanded to account for these parameters and also to provide the ability to declare more than one connection to the same instance of Microsoft SQL Server.

Microsoft SQL Server Cursors With ODBC

In earlier, ODBC-based releases of the adapter, the SET parameter CONCUR is used to control the ODBC cursor type. The CONCUR parameter has two settings: RONLY results in client-side cursors, and ROWVER forces cursors to be server-side. Release 5.2 of the adapter takes advantage of OLE DB's ability to directly control the cursor type via a new CURSOR parameter. The SET parameter CURSORS has two settings: CLIENT and SERVER. By default, the adapter assigns cursor properties that provide maximum performance for the given environment.

Mapping of UNIQUEIDENTIFIER and BIT Data Types

OLE DB maps the following data types differently than ODBC:

- UNIQUEIDENTIFIER is mapped to 38 characters. ODBC maps this data type to 36 characters. This difference results from the fact that OLE DB encloses the value in braces ({}).
- BIT is mapped to -1 for TRUE and 0 for FALSE. ODBC maps TRUE to 1.

CHAPTER 20

Getting Started in Microsoft SQL Server Analytical Engine

Topics:

- Preparing the Microsoft SQL Server Analytical Engine Environment
- Configuring the Data Adapter for Microsoft SQL Server Analytical Engine
- Managing Microsoft SQL Server Analytical Engine Metadata
- Customizing the Microsoft SQL Server Analytical Engine Environment

The Data Adapter for Microsoft SQL Server Analytical Engine allows applications to access Microsoft SQL Server Analytical Engine data sources. The adapter converts application requests into native Microsoft SQL Server Analytical Engine statements and returns optimized answer sets to the requesting application.

Preparing the Microsoft SQL Server Analytical Engine Environment

The Data Adapter for Microsoft SQL Server Analytical Engine minimally requires the installation of the Microsoft OLE DB software which is part of the MDAC (Microsoft Data Access Components) installation.

Procedure How to Set Up the Environment on Windows NT/2000

On Windows NT/2000, you only need to install MDAC 2.6 or later.

The Data Adapter for Microsoft SQL Server Analytical Engine provides read-only access to analytical data stored in Analytical Engine cubes. The data adapter is an OLE DB for OLAP consumer. It employs Multi-dimensional Data Retrieval (MDX) language to access aggregated or rolled-up data used for decision-support processing.

You can use any server front-end technology with the WebFOCUS reporting engine to access the OLAP data retrieved by the server for Windows NT/2000. This makes your OLAP data available to your entire enterprise. Additionally, data from Analytical Engine cubes can be joined with data from any other supported data source, providing additional information to your analytical process.

Setting Microsoft SQL Server Analytical Engine Security

There is only one method by which you can be authenticated when connecting to an Microsoft SQL Server Analytical Engine:

Operating System (Trusted Mode). The user ID and password used to log on to the operating system are automatically used to connect to the Analytical Engine. The server data access agent impersonates an operating system user according to the server deployment mode. The agent process establishes a connection to a Analytical Engine based on the impersonated operating system user credentials.

The Data Adapter for Microsoft SQL Server Analytical Engine recognizes the security restrictions that the OLAP Database Administrator specifies and applies them to catalogs and cubes using Analytical Engine Services. This includes the role of security on the database and the application level. Although the Server for Windows NT/2000 establishes the connection to the Analytical Engine, an impersonation of the client is used to maintain access rights.

- **Security off.** If the server is running with security off, the user ID and password of the interactive user (the person logged onto a Windows NT/2000 Server or Workstation) is passed to the Analytical Engine in order to establish a connection and access rights. This is the logon ID specified for the Windows NT/2000 Server or Windows NT/2000 Workstation after it is first started.
- **Security on.** If the server is running with security on, the user ID and password of the client connecting to the server is passed to the Analytical Engine in order to establish a connection and access rights. This process is called impersonation.

Accessing a Microsoft SQL Server Analytical Engine Server

Using the standard rules for deploying OLE DB, the server supports connections to a:

- Local Microsoft SQL Server Analytical Engine.
- Remote Microsoft SQL Server Analytical Engine. To connect to a remote Analytical Engine, it is recommended that you have MDAC 2.6 or greater installed.

Configuring the Data Adapter for Microsoft SQL Server Analytical Engine

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

The SET CONNECTION_ATTRIBUTES command allows you to declare a connection to one Microsoft SQL Server Analytical Engine.

You can declare connections to more than one Analytical Engine by issuing multiple SET CONNECTION_ATTRIBUTES commands. The actual connection takes place when the first query referencing that connection is issued (see *Changing the Default Connection* on page 20-4). You can include SET CONNECTION_ATTRIBUTES commands in an RPC or a server profile. The profile can be encrypted.

If you issue multiple SET CONNECTION_ATTRIBUTES commands, the first SET CONNECTION_ATTRIBUTES command sets the default Microsoft SQL Server Analytical Engine to be used.

Syntax How to Declare Connection Attributes Manually

For Trusted mode authentication

```
ENGINE [MSOLAP] SET CONNECTION_ATTRIBUTES datasource_name/
```

where:

MSOLAP

Indicates the Data Adapter for Microsoft SQL Server Analytical Engine. You can omit this value if you previously issued the SET SQLENGINE command.

datasource_name

Is the server name used as a connect descriptor to an Analytical Engine across the network. If omitted, the local database server will be set as the default.

Reference Declaring Connection Attributes From the Web Console

The Data Adapter for Microsoft SQL Server Analytical Engine configuration screen displays the following fields:

Field	Description
Server	Enter the name of the machine where Microsoft Analysis Services is running.

The value for this field will be used to describe connection attributes for the Microsoft SQL Server Analytical Engine.

For details on how to declare connection attributes through the Web Console, see the *Server Configuration and Operation* manual for your platform.

Changing the Default Connection

Once all Analytical Engine connections to be accessed have been declared using the SET CONNECTION_ATTRIBUTES command, there are two ways to select a specific Analytical Engine connection from the list of declared connections:

- You can select a default connection using the SET DEFAULT_CONNECTION command. If you do not issue this command, the connection_name value specified in the *first* SET CONNECTION_ATTRIBUTES command is used.
- You can include the CONNECTION= attribute in the Access File of the table specified in the current SQL query. When you include a connection name in the CREATE SYNONYM command from the Web Console, the CONNECTION= attribute is automatically included in the Access File. This attribute supercedes the default connection.

Syntax **How to Change the Default Connection**

```
ENGINE [MSOLAP] SET DEFAULT_CONNECTION [datasource_name]
```

where:

MSOLAP

Indicates the Data Adapter for Microsoft SQL Server Analytical Engine. You can omit this value if you previously issued the SET SQLENGINE command.

datasource_name

Is the data source name specified in a previously issued SET CONNECTION_ATTRIBUTES command. If omitted, then the local database server will be set as the default. If this connection name has not been previously declared, a FOC1671 message is issued.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection_name specified in the last command will be the active connection_name.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, a FOC1671 message is issued.

Example **Changing the Default Connection**

The following SET DEFAULT_CONNECTION command selects the Microsoft SQL Server Analytical Engine named OLAPSERV as the default Analytical Engine:

```
ENGINE MSOLAP SET DEFAULT_CONNECTION OLAPSERV
```

Note: You must have previously issued a SET CONNECTION_ATTRIBUTES command for OLAPSERV.

Managing Microsoft SQL Server Analytical Engine Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Microsoft SQL Server Analytical Engine data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Analytical Engine cube that is accessible from a server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

For details on how to create a synonym through the Web Console, see the *Server Configuration and Operation* manual for your platform.

Syntax

How to Create a Synonym Manually

```
CREATE SYNONYM appname/synonym FOR cube_name DBMS MSOLAP [AT  
datasource_name ]  
DATABASE database_name  
END
```

where:

appname

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, then this application name must not be used.

synonym

Is an alias for the data source (it can be a maximum of 64 characters).

cube_name

Is the name of the OLAP cube. See the Microsoft SQL Server Analytical Engine documentation for specific naming conventions.

MSOLAP

Indicates the Data Adapter for Microsoft SQL Server Analytical Engine.

AT *datasource_name*

Is the *datasource_name* as previously specified in a SET CONNECTION_ATTRIBUTES command. When the server creates the synonym, this *connection_name* becomes the value for DATASOURCE= in the Access File.

database_name

Is the name of the data source (catalog) the OLAP cube is using.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note:

- CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.
- CREATE SYNONYM creates a Master File and Access File which represents the server metadata.

Example Using CREATE SYNONYM

Use the following syntax to create a synonym for the variable "sales"

```
CREATE SYNONYM sales FOR sales DBMS MSOLAP AT OLAPSVR1
DATABASE 'FOODMART 2000'
END
```

Generated Master File sales.mas

```
FILENAME=SALES, SUFFIX=MSOLAP,$
SEGNAME=SALES, SEGTYPE=S0,$
$ DIMENSION: -MEASURES-
FIELD= PROFIT,
TITLE= 'Profit',
ALIAS= 'Profit',
USAGE= P10.2,ACTUAL= P8,MISSING= ON,$
$ DIMENSION: Customers
FIELD= COUNTRY,
TITLE= 'Country'
WITHIN= '*[Customers]',
ALIAS= 'Country',
USAGE= A6,ACTUAL= A6,MISSING= OFF,$
```

The Master File that is produced contains a segment for each dimension in the OLAP outline. For each generation within a dimension, a field description is produced. The WITHIN keyword is used to describe the hierarchy inherent in each dimension structure.

Note: Master Files should not be altered. Manipulating the Master File causes incorrect query results.

Generated Access File sales.acx

```
SEGNAME=SALES ,  
CUBENAME=SALES ,  
DATASOURCE=EDAKNW3 ,  
CATALOG=FOODMART 2000 ,  
SCHEMA= , $
```

Reference Access File Keywords

Keyword	Description
CUBENAME=	Indicates the name of the OLAP cube.
DATASOURCE=	Indicates the Microsoft SQL Server Analytical Engine name.
CATALOG=	Indicates the name of the data source (catalog) the OLAP cube is using.

Note: Access Files reflect the structure of the cube Master File that is produced and contains a segment for each dimension in the OLAP outline. For each generation within a dimension, a field description is produced. The WITHIN keyword is used to describe the hierarchy inherent in each dimension structure.

Customizing the Microsoft SQL Server Analytical Engine Environment

The Data Adapter for Microsoft SQL Server Analytical Engine provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Activating NONBLOCK Mode and Issuing a TIMEOUT Limit

You can use the NONBLOCK command to prevent runaway queries. The Data Adapter for Microsoft SQL Server Analytical Engine uses non-blocking protocol for query execution.

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to Microsoft SQL Server Analytical Engine.

Syntax How to Activate NONBLOCK Mode and Issue a TIMEOUT Limit

```
SET NONBLOCK n [TIMEOUT m]
```

where:

n

Is the polling period in seconds.

m

Is the total timeout after which the query is terminated if it does not get completed.

Accelerating Queries

Using the SLICEROPTIMIZATION command usually significantly accelerates queries where some dimensions are referenced for tests only.

Syntax **How to Use SLICEROPTIMIZATION**

```
SET SLICEROPTIMIZATION {ON|OFF}
```

where:

ON

Optimizes some queries by putting some member sets on the slicer axis and using the AGGREGATE function rather than retrieving them. This is the default setting.

OFF

Suppresses the optimization. This is provided for compatibility and performance comparison purposes.

Example **Using the SLICEROPTIMIZATION Command**

In the following request, the PRODUCT dimension is referenced for testing only, therefore the AGGREGATE of all the products containing 'Best' can be put on the slicer. As a result, much less data will have to be read by the adapter.

```
TABLE FILE SALES  
WRITE SALES  
BY YEAR  
WHERE PRODUCT_NAME LIKE 'Best%';  
END
```

CHAPTER 21

Getting Started in MODEL 204

Topics:

- Preparing the MODEL 204 Environment
- Configuring the Data Adapter for MODEL 204
- MODEL 204 Overview and Mapping Considerations
- Managing MODEL 204 Metadata
- Customizing the MODEL 204 Environment
- Adapter Tracing for MODEL 204

The Data Adapter for MODEL 204 allows applications to access MODEL 204 data sources. The adapter converts application requests into native MODEL 204 statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

Preparing the MODEL 204 Environment

Before you install the adapter, please review the following list of software requirements:

- MODEL 204 must be installed and working. If it is not, contact your MODEL 204 database administrator. The adapter may be used with MODEL 204 Version 2.2.0 and subsequent versions.
- The server must be installed on your system. If it is not, contact your server administrator or consult the appropriate server installation guide.

Configuring the Data Adapter for MODEL 204

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish. The Data Adapter for MODEL 204 is configured from the Web Console.

Procedure How to Configure the Adapter

To configure the Data Adapter for MODEL 204:

1. Edit the ISTART JCL:
 - Allocate the MODEL 204 load library to ddname STEPLIB.
 - Any private Master File libraries and *prefix*.EDAMFD.DATA should be allocated to ddname MASTER in the JCL.
 - Any private Access File libraries and *prefix*.EDAAFD.DATA should be allocated to ddname ACCESS in the JCL.
2. From the Web Console Adapter Add screen, click *Configure*.

Specifying Account and File Passwords

Your MODEL 204 database administrator must provide account and file security information in order for the adapter to access password protected MODEL 204 files or groups.

An account consists of the MODEL 204 user ID and password. You can specify these security values by either including the ACCOUNT and ACCOUNTPASS attributes in the first declaration of the Access File, or by issuing the adapter M204IN SET M204ACCNT and M204PASS commands. Your database administrator can also provide MODEL 204 logon and account information with a security exit subroutine.

You must also know the file name and password (if there is one) for each MODEL 204 file or group you access. Specify these values with the FILE and PASS attributes in each SEGNAME declaration of the Access File.

For more information, see *Customizing the MODEL 204 Environment* on page 21-31.

Testing the Adapter Installation

To verify the adapter installation, run a simple query using one of the pairs of Master and Access Files you have defined for a MODEL 204 file.

MODEL 204 Overview and Mapping Considerations

A MODEL 204 file can represent one file or a collection of files called a group. A single MODEL 204 file can contain records that vary in length and format; a record can consist of fields that also vary in length. Fields are the smallest elements or units of data. They can occur more than once per record. A unit of records with identical sets of fields is called a logical record type. A MODEL 204 file is defined in a MODEL 204 file description.

In the sections that follow, you will notice that:

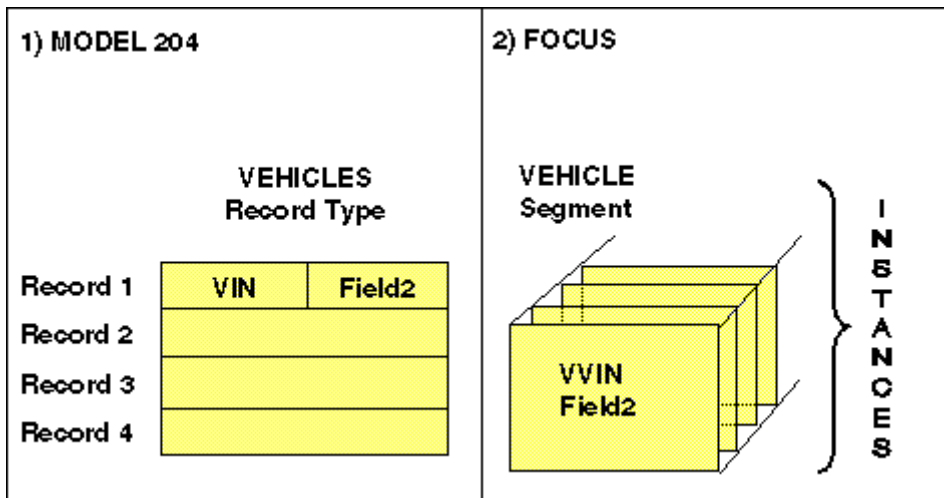
- A MODEL 204 field is equivalent to a server Master File field.
- A logical record type corresponds to a server Master File segment. The individual record corresponds to a segment instance.
- MODEL 204 fields that appear more than once in a record map to a server Master File OCCURS segment.
- One or more record types can be described in any order in a pair of Master and Access Files.
- One or more MODEL 204 files can be described in a pair of Master and Access Files.

Files With One Logical Record Type

A MODEL 204 file that contains records with identical sets of fields (the same logical record type) can be represented as a single Master File segment. Each MODEL 204 field that occurs only once becomes an Master File field. The ALIAS attribute in the Master File identifies the MODEL 204 field name.

In the Master File, describe the logical record type as a segment; the segment description can include some or all of the MODEL 204 fields in any order. The corresponding Access File identifies the record type's MODEL 204 file or group name and its password; the Access File can also include FIELD declarations that specify a TYPE attribute to indicate the appropriate suffix operators for MODEL 204 key fields. Master File field suffix operators (for example, KEY) identify different types of MODEL 204 key fields.

Note: A MODEL 204 field that can occur more than once should be represented as a separate OCCURS segment.



1. Is one MODEL 204 file with one logical record type.
2. Is the equivalent server Master File segment. The Master File field VVIN represents the MODEL 204 field VIN.

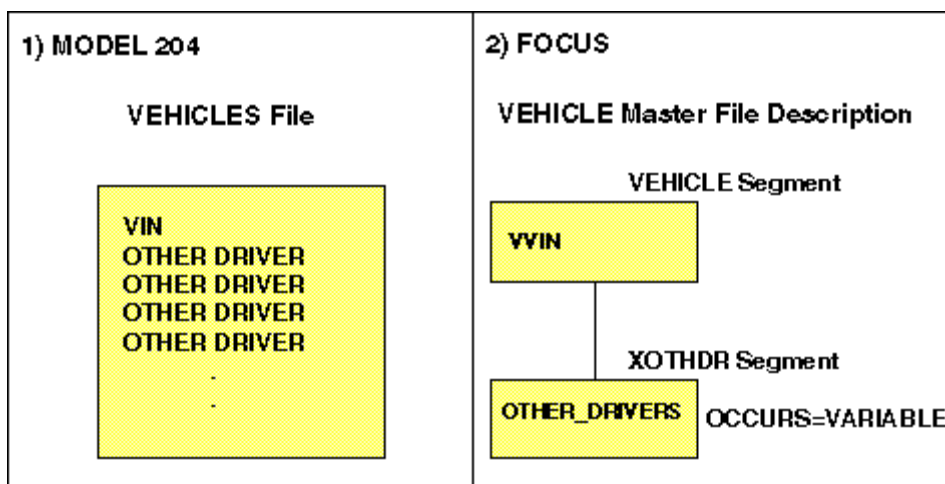
Files With Multiply Occurring Fields

In MODEL 204 files, some fields may appear more than once per record. Represent each multiply occurring field separately from the non-repeating fields as a server Master File OCCURS segment. In this case, the Master File contains a parent segment to describe all the non-repeating fields, and a separate dependent OCCURS segment for each multiply occurring field.

When you create a Master File, describe the OCCURS segment only if you need the multiply occurring field for your reports; you are not required to describe every multiply occurring field. Do not describe the OCCURS segment in the corresponding Access File unless its size exceeds the buffer capacity, a situation that is explained in *Customizing the MODEL 204 Environment* on page 21-31.

Describing Files to the Server

The following diagram illustrates one multiply occurring field and its equivalent OCCURS segment:



1. Is the MODEL 204 VEHICLES file with one logical record type and one multiply occurring field called OTHER DRIVER that identifies drivers besides the principal driver.
2. Is the equivalent multi-segment server structure. The parent (or root, in this case) is the VEHICLE segment; it contains all of the non-repeating fields. The dependent segment is the OCCURS segment which contains iterations of the MODEL 204 OTHER DRIVER field. In the Master File, the segment declaration for the dependent segment must include the OCCURS attribute.

If a MODEL 204 file contains multiple logical record types, each logical record type corresponds to one segment. You can represent a MODEL 204 file with several logical record types either:

- As a multi-segment server Master File structure with a hierarchical retrieval path. The segments are cross-referenced by Embedded Joins defined on the parent/child relationships.
- As several individual segments, each described in a separate pair of Master and Access Files.

In a multi-segment Master File, you can include segment descriptions for all of the logical record types or only those you need for your reports. To identify the parent of a dependent segment, specify the PARENT attribute in the segment declaration of the dependent segment.

In the corresponding Access File, for each logical record type you must specify:

- The MODEL 204 file and password.
- The MODEL 204 record type field and the unique value that identifies it.
- The shared field that implements the Join relationship.
- MODEL 204 key fields (TYPE attributes in FIELD declarations identify the appropriate suffix operators).

Note: Record type fields identify individual record types that reside in the same MODEL 204 file. MODEL 204 files that consist of one logical record type do not require record type fields. In the Access File, the RECTYPE and RECTVAL attributes specify record type fields and the values.

Mapping MODEL 204 and Server Relationships

In MODEL 204, interfile relationships are not coded in the MODEL 204 file description. This provides a certain degree of flexibility because the MODEL 204 files can be cross-referenced dynamically. On the other hand, the burden of connecting the MODEL 204 files falls on the user.

MODEL 204 relationships between logical record types are cross-referenced using shared key fields. The adapter supports this implementation. You can logically join MODEL 204 files using the adapter with either of the following techniques:

- Issue JOIN commands to dynamically join the MODEL 204 files.
- Create a multi-segment server Master File structure and describe the Join relationships in a pair of Master and Access Files (this multi-segment structure is also called an Embedded Join or Server View).

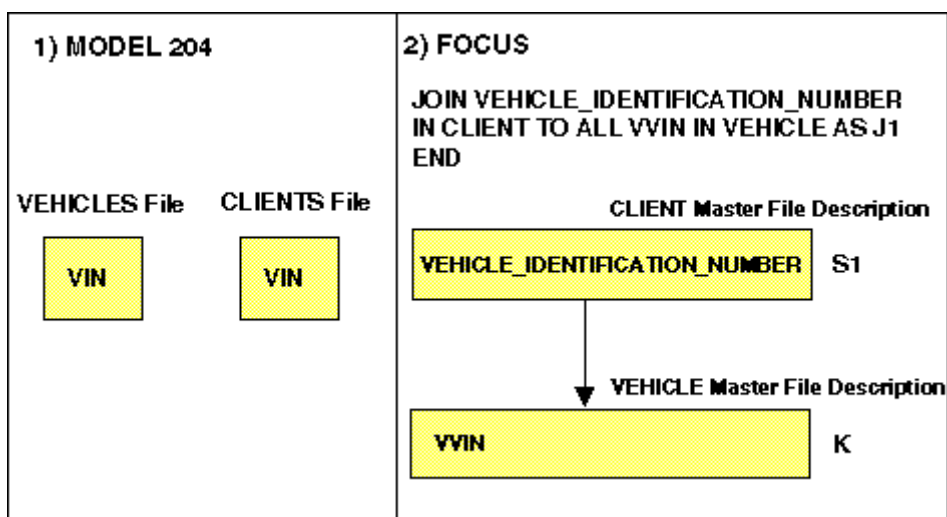
Both techniques are described in the following sections.

Dynamic Joins

If there is a shared or common field, you can use the server JOIN command to dynamically Join:

- Separate physical MODEL 204 files that each contain one logical record type. These MODEL 204 files are usually described as single segment Master Files.
- Logical record types that are described to the server as single segment Master Files, but that originate from the same MODEL 204 file.
- Multi-segment server structures that represent several MODEL 204 files and/or different record types from the same MODEL 204 file. Multi-segment Master Files are explained in *Embedded Joins* on page 21-9.

The following diagram illustrates a MODEL 204 Join and its equivalent Server Join:

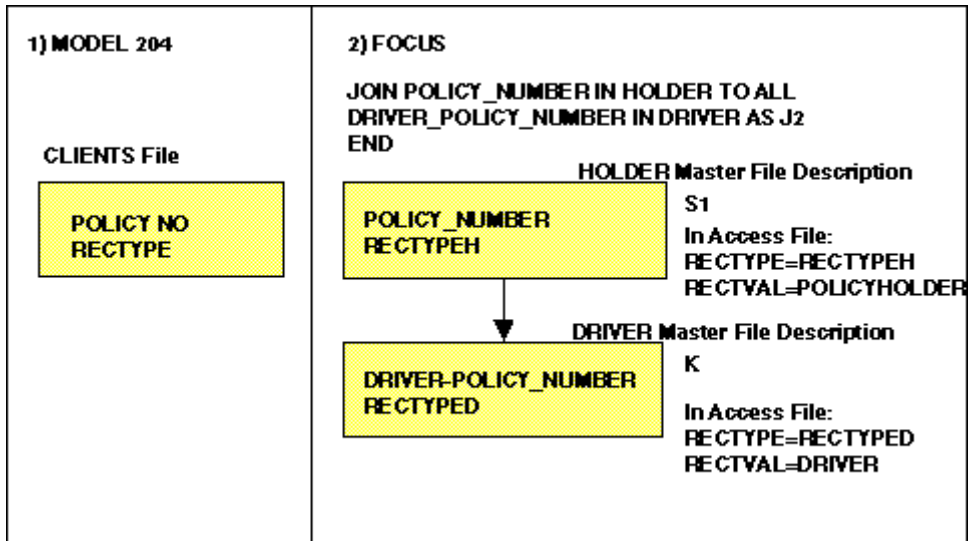


1. Are two MODEL 204 files, VEHICLES and CLIENTS. Each file contains one logical record type for this example. The common or key field is VIN.
2. Are two equivalent server segments. Each segment is described in a separate pair of Master and Access Files. The field names for the common field are VEHICLE_IDENTIFICATION_NUMBER and VVIN. The MODEL 204 key fields are specified with the TYPE=KEY attribute in the Access File.

The JOIN command includes the names of the Master Files and the common field names. To retrieve data from the joined structure, issue a report request that specifies the host (or parent) Master File from the JOIN command.

Note: Since in this example each MODEL 204 file consists of one logical record type, record type fields are not specified in the Access Files.

The following diagram illustrates a MODEL 204 file with two record types and the Server Join:



1. Is the MODEL 204 CLIENTS file; it contains two logical record types. The common or key field is POLICY NO. The RECTYPE field contains the value POLICYHOLDER for the HOLDER record type and the value DRIVER for the DRIVER record type.
2. Are two equivalent Master File segments. Each segment is described in a separate pair of Master and Access Files. The field names for the common field are POLICY_NUMBER and DRIVER_POLICY_NUMBER. The MODEL 204 key fields are specified with the TYPE=KEY attribute in the Access File. Since both record types reside in the same MODEL 204 file, each Access File segment declaration also includes RECTYPE (record type) and RECTVAL (record type value) attributes.

The JOIN command includes the names of the Master Files and the common field names. To retrieve data from the joined structure, issue a report request that specifies the host (or parent) Master File from the JOIN command.

Embedded Joins

You can also implement Join relationships by describing a multi-segment FOCUS structure in a single pair of Master and Access Files. If there is a shared or common field, you can create an Embedded Join for:

- Separate physical MODEL 204 files that each contain one logical record type. The MODEL 204 files are described as segments in the multi-segment structure.
- Logical record types that originate from the same MODEL 204 file. These record types are described as segments in the multi-segment structure. In the Access File, the RECTYPE and RECTVAL attributes in each segment record identify the record type field and its corresponding value.
- Other multi-segment FOCUS structures. A Master File might relate two hierarchical structures as one. For example, the AUTO Master File might incorporate the VEHICLE and CLIENT Master Files.

Embedded Joins offer two advantages:

- They create a logical FOCUS view of the record types. The view may also provide a measure of security by limiting access to segments or fields.
- Users do not have to issue an explicit JOIN command.

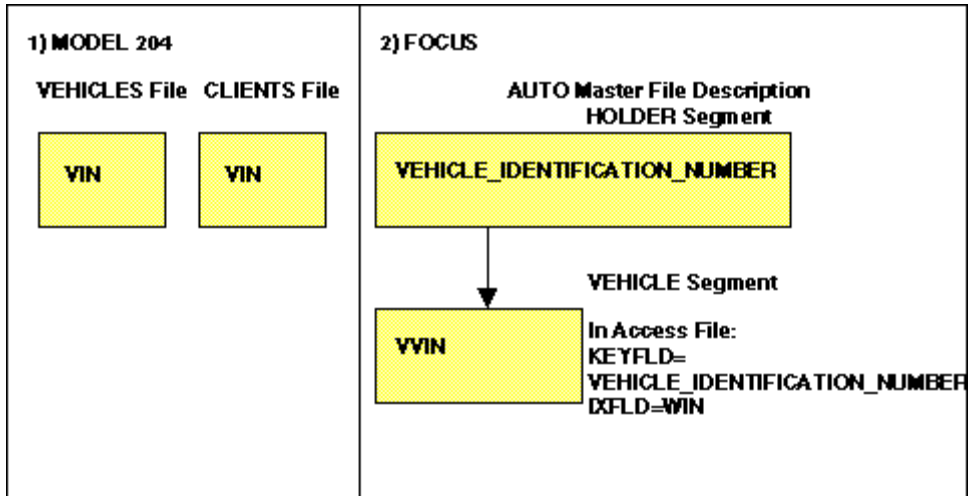
When you create a Master File, describe the MODEL 204 logical record types or files as a hierarchy. Start first by choosing a record type or file to be the root segment. The root segment has dependent segments which, in turn, may have dependent segments. The dependent segments identify the superiors (or parents) with the PARENT attribute. Any record type can act as a dependent provided that its shared field is a key field.

In the associated Access File, for each pair of related segments, specify the shared field from the parent and dependent segments with the KEYFLD and IXFLD attributes. The Interface implements the relationship by matching values at run time.

The Embedded Join is executed automatically when you issue a report request that references fields from two related segments. You do not specify the shared field in your report request; the selection of the shared field is transparent.

Note: It is possible to join unrelated MODEL 204 files and record types that do not have common fields by describing a DUMMY segment.

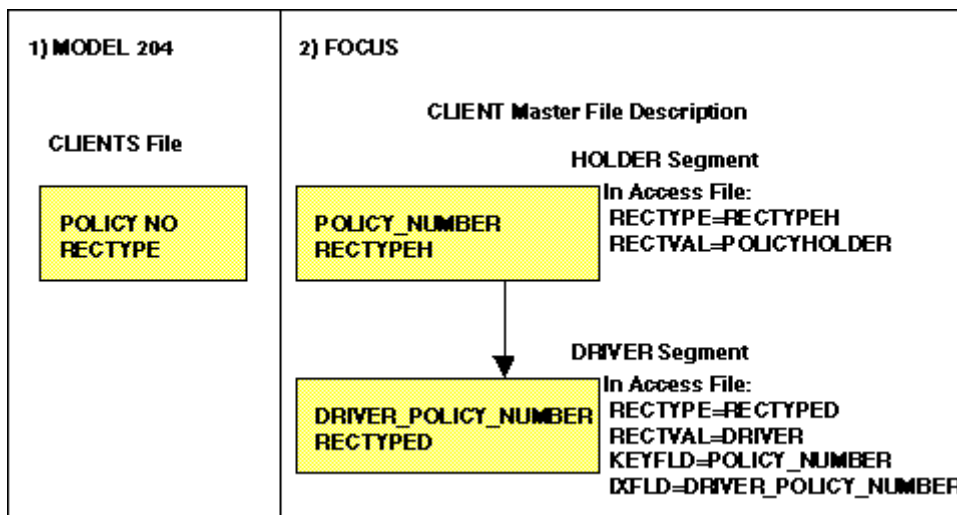
The following diagram illustrates an Embedded Join for two MODEL 204 files:



1. Are two MODEL 204 files, VEHICLES and CLIENTS. Each file contains one logical record type for this example. Since they exist as physically separate files, they do not contain a MODEL 204 record type field. The common or key field is VIN.
2. Are two equivalent FOCUS segments described in one pair of Master and Access Files. The field names for the common field are VEHICLE_IDENTIFICATION_NUMBER and VVIN. The MODEL 204 key fields are specified with the TYPE=KEY attribute in the Access File.

To create a parent/child relationship, describe the HOLDER segment as the parent (or root) and the VEHICLE segment as the dependent in the Master File. Include the PARENT=HOLDER attribute in the segment declaration for VEHICLE. In the Access File, specify the KEYFLD and IXFLD attributes to create the Embedded Join.

The following diagram illustrates an Embedded Join for a MODEL 204 file with two record types:



1. Is the MODEL 204 CLIENTS file; it contains two logical record types. The common or key field is POLICY NO. The RECTYPE field contains the value POLICYHOLDER for the HOLDER record type and the value DRIVER for the DRIVER record type.
2. Are two equivalent FOCUS segments described in one pair of Master and Access Files. The field names for the common field are POLICY_NUMBER and DRIVER_POLICY_NUMBER. The MODEL 204 key fields are specified with the TYPE=KEY attribute in the Access File. Since both record types reside in the same MODEL 204 file, the Access File segment declarations also include the RECTYPE (record type) and RECTVAL (record type value) attributes.

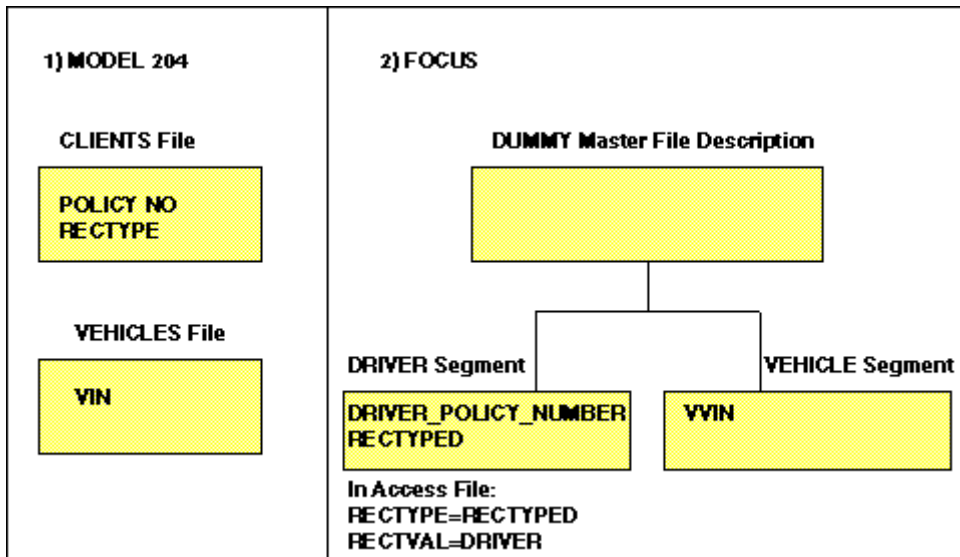
To create a parent/child relationship, describe the HOLDER segment as the parent (or root) and the DRIVER segment as the dependent in the Master File. Include the PARENT=HOLDER attribute in the segment declaration for DRIVER. In the Access File, specify the KEYFLD and IXFLD attributes in the DRIVER segment declaration to create the Embedded Join.

Joining Unrelated Files

You can describe unrelated MODEL 204 files or logical record types as Embedded Joins even if a shared field does not exist. To do so, describe a special dummy segment (SEGNAME=DUMMY) as the root in the Master File. Then, specify the unrelated segments as parallel dependents of the dummy root. Since the dummy segment is a virtual segment, do not describe fields for it in the Master File, and do not include a corresponding segment declaration for it in the Access File.

If your report request includes fields from two or more segments, FOCUS makes only one retrieval pass over the answer set returned by the MODEL 204 DBMS. Fields specified for sort phrases (BY and ACROSS) and screening tests (IF and WHERE) must lie on the same retrieval path as the other requested fields. You cannot use a field from one unrelated segment to sort the data in other segments.

The following diagram illustrates two MODEL 204 files, the CLIENTS file and the VEHICLES file. These two files are joined using a FOCUS dummy segment:



1. Are two MODEL 204 files, the CLIENTS file and the VEHICLES file. The record type field in the DRIVER segment identifies all client records that have the value 'DRIVER'. A dummy segment is needed because there is no shared field to cross-reference or join these files.
2. Is an equivalent multi-segment FOCUS structure. The root is a DUMMY segment. The two dependents are related to the root; they are not related to each other because a common field does not exist.

To create a Master File for this structure, specify the segment declaration for the dummy segment (SEGNAME=DUMMY) first, so it functions as the root. Do not describe fields for it. Then, describe the dependent segments, and include the PARENT=DUMMY attribute to identify the dummy segment as the parent. In the associated Access File, do not specify a segment declaration for the dummy segment; include segment declarations only for the dependent segments. The record type field is required in the DRIVER segment since the DRIVER segment comes from a MODEL 204 file, CLIENTS, that contains two logical record types. Do not specify KEYFLDIXFLD values.

Summary of Mapping Rules

A careful examination reveals that MODEL 204 and FOCUS use similar elements:

MODEL 204 Entities =	FOCUS Entities
A MODEL 204 field	A FOCUS field
A logical record type	A FOCUS segment
An individual record	A segment instance
A MODEL 204 file with one record type	A FOCUS segment
Multiply-occurring MODEL 204 fields in a record	A FOCUS OCCURS segment

Some general rules for describing MODEL 204 files are:

- Each logical record type becomes a FOCUS segment. A FOCUS segment may represent an entire MODEL 204 file (with one logical record type) or one logical record type from a MODEL 204 file with several record types.
- A MODEL 204 file containing two or more logical record types must have a record type field to identify each record type.
- A MODEL 204 file containing one logical record type does not require a record type field; the MODEL 204 file may contain an optional one.
- Relationships can be implemented with the JOIN command or by describing Embedded Joins in the FOCUS file descriptions. The maximum number of MODEL 204 files or logical record types that can be related is 64 for Embedded Joins and 16 for Dynamic Joins.

Managing MODEL 204 Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a Master File and Access File that describe the structure of the data source and the server mapping of the MODEL 204 data types. The logical description of an MODEL 204 file is stored in a Master File, which describes the field layout. The physical attributes of the MODEL 204 file, such as the database number, file number, descriptors, and security are stored in the Access File.

Master Files

A MODEL 204 file consists of records and fields. It can represent one file or a collection of files called a group. If records have identical sets of fields, the unit of records is called a logical record type. A logical record type is described as a single segment in the Master File.

A Master File consists of file, segment, and field declarations. Rules for declarations are:

- Each declaration must begin on a separate line and be terminated by a comma and dollar sign (,\$). Text appearing after the comma and dollar sign is treated as a comment.
- A declaration can span as many lines as necessary as long as no attribute=value pair is separated. Commas separate attribute=value pairs.
- Certain attributes are required; the rest are optional.
- The following sections summarize the syntax for each type of declaration and then present detailed explanations for each attribute.

Syntax

How to Declare File Attributes

Each Master File begins with a file declaration that names the file and describes the type of data source—a MODEL 204 file or group in this case. The file declaration has two attributes.

`FILENAME (FILE)`

Identifies the Master File.

`SUFFIX`

Identifies the data adapter needed to interpret the request.

The syntax is

`FILE[NAME]=name , SUFFIX=M204IN [, $]`

where:

name

Is a 1 to 8 character name that complies with server file naming conventions.

`M204IN`

Is the SUFFIX value that specifies the Data Adapter for MODEL 204.

Syntax **How to Declare Segment Attributes**

Each logical record type described in a Master File requires a segment declaration that consists of at least two attributes: SEGNAME and SEGTYPE.

The syntax is

```
SEGNAME= name | DUMMY , SEGTYPE= S | U [ , PARENT=parent ] , $
```

where:

name

Is a 1 to 8 character name that identifies the segment. The value DUMMY creates a virtual segment used for joining unrelated segments.

S | U

Indicates to the adapter that the MODEL 204 DBMS handles the storage order of the data. The default setting is S.

parent

For multi-segment structures (Embedded Joins) only, is the SEGNAME value for the parent record type. This attribute is required in segment declarations that describe dependent segments.

SEGNAME

The SEGNAME (or SEGMENT) attribute identifies each MODEL 204 logical record type. The SEGNAME value can be a maximum of 8 alphanumeric characters. It may be the name of the record type or an arbitrary name and must be unique within a Master File.

The SEGNAME value from the Master File is also specified in the Access File where the segment declaration identifies the name of the MODEL 204 file. In this manner, the SEGNAME value serves as a link to the actual MODEL 204 file name.

SEGTYPE

In a single segment Master File, the SEGTYPE value is S. The MODEL 204 DBMS handles the storage order of the data.

In a multi-segment Master File, SEGTYPE values for dependent segment declarations can be S or U (for unique).

- SEGTYPE=S indicates that the dependent record type has a one-to-many or many-to-many relationship with the record type named as its parent. For every record of the parent record type, there may be more than one record in the dependent record type.
- SEGTYPE=U indicates that the dependent record type has a one-to-one or a many-to-one relationship with the record type named as its parent. For every record of the parent record type, there may be (at most) one record in the dependent record type. For a one-to-one relationship to exist, both record types must share a common key. For a many-to-one relationship to exist, the common key of the dependent record type must be a subset of the common key of the parent record type.

If the SEGTYPE attribute is omitted from any segment record in the Master File, its value defaults to S.

PARENT

The PARENT attribute is required in dependent segment declarations. Its value is the SEGNAME value of the dependent record type's parent (the logical record type to which it will be related at run time).

Syntax **How to Declare Field Attributes**

Each record in a logical record type consists of one or more fields. These MODEL 204 fields are described in the Master File with the primary attributes FIELDNAME, ALIAS, USAGE, and ACTUAL. You are only required to describe MODEL 204 fields you use in reports, and you can specify them in any order within each segment.

The syntax for a field declaration is

```
FIELD[NAME]=field, [ALIAS=] [m204field], [USAGE=] display, [ACTUAL=] An
, $
```

where:

field

Is a 1 to 66 character field name. In requests, the field name can be qualified with the Master File and/or segment name; although the qualifiers and qualification characters do not appear in the Master File, they count toward the 66 character maximum.

m204field

Is the MODEL 204 field name, which can be up to 66 characters (or, optionally left blank if the field name exceeds 66 characters and is described in the Access File).

display

Is the server display format for the field.

An

Is the server definition of the MODEL 204 field format and length (n).

You can omit the ALIAS, USAGE, and ACTUAL keywords from the field declaration if the values are specified in the standard order (FIELD, ALIAS, USAGE, ACTUAL). For example, the following declarations are equivalent:

```
FIELD = YEAR, ALIAS=YEAR, USAGE=A2, ACTUAL=A2, $
FIELD = YEAR, YEAR, A2, A2, $
```

FIELDNAME

Field names must be unique within the Master File and may consist of up to 66 alphanumeric characters. MODEL 204 field names are acceptable values if they meet the following naming conventions:

- Names can consist of letters, digits, and underscore characters. Special characters and embedded blanks are not advised.
- The name must contain at least one letter.

Since the field name appears as the default column title for reports, select a name that is representative of the data. You can specify field names, aliases, or a unique truncation of either in requests.

Duplicate field names (the same field names and aliases) within a segment are not permitted. Requests can qualify duplicate field names from different segments with the name of the Master File and/or segment.

Note: Field names specified in OCCURS segments may not be qualified.

ALIAS

The ALIAS value for each field must be the full MODEL 204 fieldname; the adapter uses it to generate HLI calls. The ALIAS name must be unique within the segment. If the name is longer than the 66 character maximum, leave the ALIAS attribute blank in the Master File and specify the alias in the Access File instead. The ALIAS name must comply with the field naming conventions listed previously. If the MODEL 204 field name contains an embedded blank, enclose the name in single quotation marks.

Aliases may be duplicated within the Master File if they are defined for different MODEL 204 logical record types; however, the corresponding field names must be unique. For example, the ALIAS values for two MODEL 204 record type fields may be RECTYPE, but the field names must be unique.

Note: Do not use RECTYPE as a field name.

USAGE

The USAGE attribute indicates the display format of the field. An acceptable value must include the field type and length and may contain edit options. The server uses the USAGE format for data display on reports. All standard server USAGE formats (A, D, F, I, P) are available.

ACTUAL

The ACTUAL attribute indicates the server representation of MODEL 204 field formats as returned by the MODEL 204 DBMS in the data buffer. Specify an alphanumeric format (A) with a length (n) equal to the maximum length for the MODEL 204 field. If the ACTUAL length is less than the maximum length for the MODEL 204 field, field values will be truncated.

Note: Since MODEL 204 file descriptions do not define field lengths, ask your MODEL 204 database administrator or consult other sources, such as data dictionaries, for field information.

Syntax

How to Declare OCCURS Attributes

Describe each multiply occurring field that exists in a MODEL 204 record as an OCCURS segment. The segment declaration for an OCCURS segment consists of the SEGNAME, PARENT, and OCCURS attributes. It contains one field declaration to represent the multiply occurring field, and, optionally, a field declaration for an internal counter, the ORDER field. Each OCCURS segment is described as a dependent segment; its parent segment describes all non-repeating fields from the logical record type. You do not have to describe an OCCURS segment if you do not plan to use the multiply occurring field in report requests.

The syntax for an OCCURS segment declaration is

```
SEGNAME=segname , PARENT=parname , OCCURS=VARIABLE , $
```

where:

segname

Is the 1 to 8 character name of the OCCURS segment.

parname

Is the SEGNAME value of the parent segment that contains the non-repeating fields from the logical record type.

VARIABLE

Indicates that the number of occurrences varies.

Note:

- The SEGTYPE attribute is always S for OCCURS segments and, therefore, can be omitted.
- OCCURS segments generally do not require corresponding segment declarations in the Access File. An exception is for processing OCCURS segments that exceed the buffer capacity.
- Field names in OCCURS segments may not be qualified with Master File or segment names, but the TYPE attribute can define suffix operators in the Access File.

Syntax **How to Track a Sequence Within Multiply Occurring Fields**

The ORDER field is an optional counter that you can define to identify the sequence number of each multiply occurring field when the order of data is significant. You can use its value in subsequent report requests. The ORDER field does not represent an existing MODEL 204 field; it is used only for internal processing. The ORDER field must be the last field described in the OCCURS segment. The syntax is

```
FIELD=name , ALIAS=ORDER , USAGE=In , ACTUAL=I4 , $
```

where:

name

Is any meaningful name.

ALIAS

Must be ORDER.

USAGE

Is an integer (*In*) format.

ACTUAL

Must be I4.

The following annotated example illustrates an OCCURS segment in the VEHICLE Master File.

1.

```
SEGNAME=VEHICLE , $  
    FIELD=VVIN , ALIAS=VIN , A10 , A10 , $  
    FIELD=YEAR , ALIAS=YEAR , A2 , A2 , $  
    FIELD=MAKE , ALIAS=MAKE , A16 , A16 , $
```
2.

```
SEGNAME=XOTHDR , PARENT=VEHICLE , OCCURS=VARIABLE , $
```
3.

```
FIELD=OTHER_DRIVERS , ALIAS='OTHER DRIVER' , A6 , A6 , $
```
4.

```
FIELD=DRVCNT , ALIAS=ORDER , I4 , I4 , $
```

Notice that:

1. Is the segment declaration for the parent segment, VEHICLE.
2. Is the OCCURS segment declaration.
3. Is the field declaration for the multiply occurring field. Field qualifiers (Master File and segment names) are not permitted, but the TYPE attribute can define a suffix operator in the Access File.
4. Is the field declaration for the ORDER field. As an internal counter field, it does not correspond to a MODEL 204 field.

Access Files

Each Master File for a MODEL 204 file must have a corresponding Access File. The name of the Access File must be the same as that of the Master File. In OS/390, the Access File PDS is allocated to ddname ACCESS in the server JCL. The Access File associates a segment in the Master File with the MODEL 204 logical record type it describes.

The Access File consists of account, segment, and field declarations. The Access File minimally identifies account and segment information and field suffix operators. It can also contain field declarations for aliases that exceed 66 characters.

In multi-segment structures, the Access File must contain a segment declaration for each logical record type described in the Master File. Segment information may include Embedded Joins and MODEL 204 record type fields. The order of segment declarations in the Access File must correspond to the order in the Master File.

The Access File consists of 80 character declarations in comma-delimited format. Rules for declarations are:

- If you wish, you can number the declarations in columns 73 through 80.
- Each declaration must begin on a separate line and be terminated by a comma and dollar sign (,\$).
- A declaration can span as many lines as necessary as long as no attribute=value pair is separated. Commas separate attribute=value pairs.
- Blank lines and leading or trailing blanks around attribute=value pairs are ignored.
- Declarations starting with an asterisk (*) in Column 1 are treated as comments.
- Values that contain commas, equal signs, dollar signs, or embedded blanks must be enclosed in single quotation marks.

For example, the VEHICLE Access File contains one segment declaration because the VEHICLE Master File contains only one logical record type:

```

FILE=VEHICLE1, SUFFIX=M204IN

SEGNAME=VEHICLE,$
FIELD=VIN ,ALIAS=VIN ,A10 ,A10 ,$
FIELD=YEAR ,ALIAS=YEAR ,A2 ,A2 ,$
FIELD=MAKE ,ALIAS=MAKE ,A16 ,A16 ,$
FIELD=MODEL ,ALIAS=MODEL ,A12 ,A12 ,$
FIELD=BODY ,ALIAS=BODY ,A4 ,A4 ,$
FIELD=COLOR ,ALIAS=COLOR ,A10 ,A10 ,$
FIELD=VEHICLE_PREMIUM ,ALIAS='VEHICLE PREMIUM' ,A8 ,A8 ,$
FIELD=COLLISION_PREMIUM ,ALIAS='COLLISION PREMIUM' ,A6 ,A6 ,$
FIELD=LIABILITY_PREMIUM ,ALIAS='LIABILITY PREMIUM' ,A12 ,A12 ,$
FIELD=LIABILITY_LIMIT ,ALIAS='LIABILITY LIMIT' ,A3 ,A3 ,$
FIELD=DEDUCTIBLE ,ALIAS=DEDUCTIBLE ,A3 ,A3 ,$
FIELD=GARAGE_LOCATION ,ALIAS='GARAGING LOCATION' ,A12 ,A12 ,$
FIELD=TRANS ,ALIAS=TRANS ,A2 ,A2 ,$
FIELD=USAGE ,ALIAS=USAGE ,A12 ,A12 ,$
FIELD=VEHICLE_USAGE_CLASS ,ALIAS='VEHICLE USE CLASS' ,A2 ,A2 ,$
FIELD=VEHICLE_RATING ,ALIAS='VEHICLE RATING' ,A1 ,A1 ,$
FIELD=OWNER_POLICY ,ALIAS='OWNER POLICY' ,A6 ,A6 ,$
FIELD=PRINCIPLE_DRIVER ,ALIAS='PRINCIPLE DRIVER' ,A12 ,A12 ,$

SEGNAME=XOTHDR, PARENT=VEHICLE, OCCURS=VARIABLE,$
FIELD=OTHER_DRIVERS ,ALIAS='OTHER DRIVER' ,A6 ,A6 ,$
FIELD=DRUCNT ,ALIAS=ORDER ,I4 ,I4 ,$

```

The following sections summarize the syntax for each type of declaration and provide detailed explanations for each attribute.

Syntax **How to Declare Account Declaration**

The account declaration indicates authorization to access MODEL 204 files. At most, one account declaration is required; if included, it must be the first declaration in the Access File. The syntax is

```
ACCOUNT=m204id, ACCOUNTPASS=password, IFAMCHNL= channel , $
```

where:

m204id

Is the 1 to 16 character name for the MODEL 204 account (user ID).

password

Is the 1 to 16 character password for the account.

channel

Is the 1 to 8 character site specific CRAM channel name. The default is IFAMPROD for OS/390.

Syntax **How to Declare ACCOUNT and ACCOUNTPASS**

The ACCOUNT and ACCOUNTPASS attributes describe MODEL 204 accounts (user IDs) and passwords; they protect MODEL 204 files from unauthorized access. The account must have authorization to read the MODEL 204 file or group. Acceptable values are 1 to 16 character values.

You can omit the ACCOUNT and ACCOUNTPASS attributes from the declaration if you specify the values with the adapter M204IN SET M204ACCNT and M204PASS commands. These SET commands also enable you to change ACCOUNT and ACCOUNTPASS values during the session.

Adapter SET commands are explained in *Customizing the MODEL 204 Environment* on page 21-31.

Syntax **How to Declare IFAMCHNL**

The IFAMCHNL attribute describes the CRAM channel name for OS/390. You must specify the CRAM channel if more than one ONLINE or IFAM2 job is running or if the CRAM channel name is not IFAMPROD. For the channel name used at your site, check with your MODEL 204 database administrator or the IFAM2 system support staff.

The account declaration in the Access File can consist of only the IFAMCHNL attribute=value pair if the account and password values are set with the adapter M204IN SET M204ACCNT and M204PASS commands.

Syntax **How to Declare Segment Declarations**

The segment declaration in the Access File establishes the link between the server Master File and the MODEL 204 file or group. Attributes that constitute the segment declaration are: SEGNAME, FILE, PASS, KEYFLD, IXFLD, RECTYPE, RECTVAL, and ACCESS. Values for SEGNAME, FILE and PASS are required; the rest apply to multi-segment structures. The ACCESS attribute applies to OCCURS segments.

The syntax for an Access File segment declaration is

```
SEGNAME=name, FILE=m204file, PASS=password  
  [,KEYFLD=pfield] [,IXFLD=cfield] [,RECTYPE=rtfield] [,RECTVAL=rtvalue]  
  ,ACCESS = {ALL | nnn[/xxx] | AUTO[/xxx]}
```

where:

name

Is the SEGNAME value from in the Master File.

m204file

Is the 1 to 8 character name for the MODEL 204 file.

password

Is the 1 to 8 character read password for the MODEL 204 file. The value can be blank if a password does not exist.

pfield

Is the field name of the common field from the parent segment, and is used to implement the Join relationship. It is required in dependent segment declarations for multi-segment structures.

cfield

Is the field name of the common field from the dependent segment, and is used to implement the Join relationship. It is required in dependent segment declarations for multi-segment structures.

rtfield

Is the field name for the MODEL 204 record type field that exists when a MODEL 204 file has several logical record types.

rtvalue

Is the value that identifies the MODEL 204 record type or a field name.

[ALL](#)
[nnnn/xxx](#)
[AUTO/xxx](#)

Indicates how OCCURS segments should be processed when all occurrences cannot fit into the buffer at once and a (FOC4883) IFMORE FAILED error condition results. These options are described *Access Files* on page 21-21.

Note:

- For multi-segment structures, the order of the Access File segment declarations must correspond to the order in the Master File.
- Dummy segments used to link unrelated segments do not have corresponding segment declarations in the Access File.

Syntax **How to Declare SEGNAME**

The SEGNAME value must be the first attribute in each Access File segment declaration; it must be identical to the SEGNAME value in the Master File.

Syntax **How to Declare FILE and PASS**

The FILE and PASS attributes are required in each Access File segment declaration; they describe MODEL 204 file security information.

The value for the FILE attribute is the name of the MODEL 204 file or group that contains the logical record type. The value for the PASS attribute is either the read password associated with the MODEL 204 file or blank if the password does not exist. For both attributes, acceptable values are 1 to 8 character values.

Access Files can be encrypted to prevent access to this security information.

Syntax **How to Declare KEYFLD and IXFLD**

The KEYFLD and IXFLD attributes identify the common fields for parent/child relationships in a multi-segment Master File. These relationships are referred to as Embedded Joins, server views, or cross-references.

For each dependent segment, the KEYFLD and IXFLD attributes identify the field names of the common or shared field that implements the Embedded Join. The parent field supplies the value for cross-referencing; the dependent field contains the corresponding value. The adapter implements the relationship by matching values at run time.

The value for the KEYFLD attribute is a 1 to 66 character field or alias name from the parent segment. The value for the IXFLD attribute is a 1 to 66 character field or alias name from the dependent segment.

Note:

- Include the pair of attributes in Access File segment declarations for dependent segments. Do not specify them in the segment declaration for the root segment.
- Do not specify KEYFLD and IXFLD attributes for dependent segments of a virtual parent segment (PARENT=DUMMY).

A JOIN can be based on more than one field in the host and cross-referenced logical record types. If the MODEL 204 file uses multiple fields to establish a relationship or link between logical record types, you can specify concatenated fields in an Embedded Join (you can also specify multiple fields with the Dynamic JOIN command).

In a multi-field Embedded Join, the KEYFLD and IXFLD values consist of a list of the component fields separated by slashes (/). Additional Access File attributes are not required. The syntax is

```
KEYFLD = field1/field2/....
```

```
IXFLD = cfield1/cfield2/....
```

where:

```
field1/...
```

Is a composite of up to 16 key fields from the parent segment. Slashes are required.

```
cfield1/...
```

Is a composite of up to 16 key fields in the dependent segment.

The adapter compares each field pair for comparable data formats prior to format conversion; it evaluates each field pair with the following rules:

- Any of the MODEL 204 key fields listed in Field Declarations can be used to create an Embedded Join.
- Parent and dependent fields must be real fields.
- All participating fields for either the parent or dependent file must reside in the same segment.
- KEYFLD and IXFLD attributes must specify the same number of fields. If the format of the parent field is longer than the format of the dependent field, its values are truncated. If the format of the parent field is shorter, its values are padded with zeros or blanks.
- The KEYFLD and IXFLD attributes can consist of a maximum of 16 concatenated fields in high to low significance order. You must specify the field order; the order determines how the values will be matched.
- Each pair of fields in the KEYFLD/IXFLD attribute must have comparable data formats. The formats of the parent fields must be compatible with the formats of the dependent fields as specified in the MODEL 204 file description.

To implement Joins, the MODEL 204 DBMS converts the alphanumeric server field formats to equivalent MODEL 204 field formats in order to perform the necessary search and match operations. When the MODEL 204 DBMS returns the answer set of matched values, it also converts the values back to alphanumeric formats.

Internally, each KEYFLD value is passed to the MODEL 204 DBMS in the IFFIND call to retrieve all matching records (segment instances) for the dependent segment.

Syntax

How to Declare RECTYPE and RECTVAL

The RECTYPE and RECTVAL attributes identify the MODEL 204 record type field and its corresponding value. A MODEL 204 record type field exists in a MODEL 204 file description when the file has more than one logical record type. If the MODEL 204 file consists of one logical record type, it may contain an optional record type field.

The value for the RECTYPE attribute is the 1 to 66 character field or alias name of the MODEL 204 record type field. The value for the RECTVAL attribute is the identifying value for the record type field; it can be up to 255 characters in length or a field name. The RECTVAL attribute is required whenever the RECTYPE attribute is used.

The RECTYPE and RECTVAL attributes are required in each Access File segment declaration (including the one for the root) if the MODEL 204 file contains record types.

Note: Do not use the name RECTYPE as a field name in the Master File.

ACCESS

OCCURS segments are generally described only in the Master File and do not require corresponding segment declarations in the Access File. However, sometimes the number of occurrences retrieved for an OCCURS segment exceeds the MODEL 204 buffer capacity, causing an error condition. You can alter the processing of OCCURS segments by including a segment declaration in the Access File.

The MODEL 204 buffer is used to return data to IFAM2 applications. Buffer capacity is sometimes exceeded when both of the following conditions exist:

- A multiply occurring field repeats a large number of times (for example, 2000 occurrences).
- The total number of occurrences multiplied by the size of the OCCURS segment is greater than the size of the IFAMBS buffer (the IFAMBS size is specified as a parameter in the MODEL 204 startup procedure).

The size of the IFAMBS buffer is determined by a predefined formula:

```
IFAMBS := (LIBUFF * 7) + 284 s 32688
```

When the buffer capacity is exceeded, the MODEL 204 DBMS sends the adapter the following error message:

```
IFMORE error M204.0903 - Results too long
```

This causes the server error message:

```
(FOC4883) IFMORE FAILED
```

If you encounter this error condition, you can include a segment declaration in the Access File to alter the number of occurrences retrieved. The adapter incorporates your specification in its HLI call to the \ DBMS.

An OCCURS segment declaration in an Access File consists of two attributes, SEGNAME and ACCESS. The ACCESS attribute indicates the type of processing to be performed. The syntax is

```
SEGNAME=name, ACCESS= {ALL | nnnn[/xxx] | AUTO[/xxx]} , $
```

where:

name

Is the SEGNAME value for the OCCURS segment from the Master File.

ALL

Indicates that the adapter will attempt to retrieve all occurrences of the OCCURS segment. This is the default.

Note: If the ACCESS attribute is omitted, the ALL setting is assumed.

nnnn

Specify the largest number of occurrences that exists for the multiply occurring field. Based on the nnnn value, the adapter may manipulate the length of the QTBL buffer for the user session and then reset it to its original length.

AUTO

Indicates that the adapter will retrieve up to 5000 occurrences. If there are more than 5000 occurrences, use nnnn instead of AUTO.

/xxx

Instructs the adapter to issue multiple IFMORE calls, with each call retrieving xxx occurrences. This parameter is optional.

Append to the nnnn or AUTO parameters to indicate that you want to control the number of records to be retrieved by each IFMORE call. If xxx is larger than the maximum allowed (based on the size of the buffers used by the Data Adapter for MODEL 204), you will receive the message:

```
(FOC4873) SPECIFIED NUMBER OF INSTANCES CANNOT FIT INTO BUFFER
```

To correct the problem, decrease the xxx value and rerun the request.

Note:

- You can turn this feature on or off by defining the OCCURS segment in the Access File. OCCURS segments not described in the Access File undergo normal processing, which means that the ALL setting is assumed and all occurrences are retrieved at one time.
- If the SEGNAME attribute is specified in the Access File without the ACCESS attribute, ACCESS=ALL is assumed by default.
- Use this feature when the following error message occurs:

```
IFMORE error M204.0903 Results too long
```

Syntax**How to Declare Field Declarations**

Field declarations in the Access File provide an alternate method for describing certain MODEL 204 field characteristics. Include a field declaration if the MODEL 204 field has a MODEL 204 key designation or if the MODEL 204 field name (the ALIAS attribute) exceeds 66 characters.

Example Supporting Types of Key Fields

The MODEL 204 DBMS supports a variety of key field types. The adapter supports the MODEL 204 key designations with comparable abbreviations called suffix operators. You describe these operators with the TYPE attribute in the Access File. The actual MODEL 204 key designation determines the proper suffix operator to apply.

The following chart lists MODEL 204 key designations and the corresponding suffix operators.

MODEL 204 Key Designation	Master File Suffix Operator
Key	KEY
Key, invisible	IVK
Ordered Character	ORA
Ordered Numeric	ORN
Numeric Range	RNG
Numeric Range, invisible	IVR

Note: Master Files created in earlier server releases may contain field declarations with suffix operators for MODEL 204 key fields. While this earlier method is supported, the recommended method is to specify suffix operators in the Access File.

Syntax How to Declare Field Attributes

The syntax for an Access File field declaration is

```
FIELD=field [,ALIAS=m204field] [,TYPE=suffix] [,ATTRIBUTE=FRV] ,[ACCDATA={STR | NUM}] ,,$
```

where:

m204field

Is a MODEL 204 field name that exceeds 66 characters; 256 characters is the maximum amount.

Note: Omit the ALIAS attribute if the ALIAS attribute in the Master File already specifies the MODEL 204 field name.

suffix

Is one of the suffix operators KEY, IVK, ORA, ORN, RNG, or IVR.

FRV

This attribute value incorporates MODEL 204 FOR EACH VALUE processing on a BY field in a server COUNT request.

STR

This attribute value will generate character string HLI selection tests for MODEL 204 fields that do not have a MODEL 204 key specification (Non-Ordered, Non-KEY, Non-Numeric Range).

NUM

This attribute value will generate numeric HLI selection tests for MODEL 204 fields that do not have a MODEL 204 key specification (Non-Ordered, Non-KEY, Non-Numeric Range).

Two additional field attributes, ATTRIBUTE and ACCDATA are available:

- ATTRIBUTE will incorporate MODEL 204 FOR EACH VALUE processing on a BY field in a server COUNT request. The field defined with ATTRIBUTE=FRV must be a MODEL 204 Ordered field or a MODEL 204 KEY/FRV field.
- ACCDATA will instruct the adapter to generate either character string or numeric HLI selection tests for MODEL 204 fields that do not have a MODEL 204 key specification (Non-Ordered, Non-KEY, Non-Numeric Range).

Customizing the MODEL 204 Environment

Each Master File for a MODEL 204 file must have a corresponding Access File. The name of the Access File must be the same as that of the Master File. In OS/390, the Access File PDS is allocated to ddname ACCESS in the server JCL. The Access File associates a segment in the Master File with the MODEL 204 logical record type it describes.

The Data Adapter for MODEL 204 provides several parameters for customizing the environment and optimizing performance.

Adapter SET commands enable you to change parameters that govern its behavior. These parameters control or identify MODEL 204 account security, the size of all adapter buffers, the size of the MODEL 204 FTBL buffer, null values in reports, locked records, and thread management. To display current adapter parameter settings, issue the adapter M204IN SET ? query command. You can issue these commands from a RPC. They remain in effect for the duration of the server client task or until you change them.

The syntax for adapter SET commands includes the ENGINE environmental qualifier and the M204IN keyword. The syntax is

```
ENGINE M204IN SET command value
```

where:

command

Is a Data Adapter for MODEL 204 command.

value

Is an acceptable value for the adapter command.

Specifying a User Account and Password

Users can supply MODEL 204 accounts and passwords with the adapter M204IN SET M204ACCNT and M204PASS commands. Authorized accounts (user IDs) and passwords protect read access to MODEL 204 files or groups of files.

The M204IN SET M204ACCNT and M204PASS commands provide an alternative to supplying security information in encrypted Access Files. You can also use the M204IN SET M204ACCNT and M204PASS commands to override existing account and password values specified with the ACCOUNT and ACCOUNTPASS attributes in the Access File.

Syntax

How to Specify a User Account and Password

To specify an account, issue the following from the command level

```
ENGINE M204IN SET M204ACCNT account
```

where:

account

Is a 1 to 16 character name for an authorized MODEL 204 account (user ID).

To specify a password for an account, issue the following from the command level

```
ENGINE M204IN SET M204PASS password
```

where:

password

Is a 1 to 16 character password for the account.

Note: If you do not issue the SET commands or you leave the values blank, they default to the values of the ACCOUNT and ACCOUNTPASS attributes specified in the Access File.

Specifying the Capacity of Adapter Buffers

Users can set or change the maximum size of all adapter buffers with the M204IN SET MAXMBUFF command.

Before you issue the M204IN SET MAXMBUFF command, consider that the MAXMBUFF value:

- Should be greater than the larger of two MODEL 204 buffer settings, LIBUFF and LOBUFF. To identify which buffer is larger, check the parameters in the MODEL 204 start-up procedure or use the Adapter Trace Facility (SET TRACEON=M204IN) to display the settings.
- Should be equal to the size of the largest segment in the Master File if the segment size is greater than the MODEL 204 LIBUFF and LOBUFF buffer settings. Having the maximum buffer size equal to that of the largest segment prevents SOC4 ABEND conditions.

Syntax How to Specify the Capacity of Adapter Buffers

To specify the capacity of the adapter buffers, issue the following

```
ENGINE M204IN SET MAXMBUFF nnnn
```

where:

nnnn

Is the maximum size in bytes. For example, specify 4096 for a MAXMBUFF value of 4K. Unless you explicitly issue this command, it is not used and there is no default value.

Controlling the Size of the FTBL Buffer

The M204IN SET FTBL command enables users to control the size of the MODEL 204 FTBL buffer. The LFTBL parameter, specified in the MODEL 204 start-up procedure, governs the size of the FTBL buffer; you can change it for the current client session only. Issue this command when a MODEL 204 error message indicates that the LFTBL value is too small.

Syntax **How to Control the Size of the FTBL Buffer**

The syntax is

```
ENGINE M204IN SET FTBL nnnn
```

where:

nnnn

Is the FTBL size in bytes. For example, specify 4096 for an FTBL value of 4K.

You should set FTBL to a value larger than the previously defined MODEL 204 LFTBL value.

Note: The FTBL value you set applies for the current user session only; it is reset when the session is ended.

Indicating Missing Data on Reports

With the adapter M204IN SET MISSING command, users can control the display of MODEL 204 null data on reports. Null data is translated into the server missing data display value. The default NODATA display value is the period (.). When the adapter MISSING attribute is set to ON, the edit specification for all IFGET calls includes an (L) format code. If MODEL 204 returns a zero in the first byte, the adapter considers that field to be missing.

Note: IFFIND calls contain IS PRESENT or IS NOT PRESENT criteria only when the request includes a server IF or WHERE MISSING selection test.

Syntax **How to Indicate Missing Data on Reports**

If you want to indicate missing data on reports, issue the following:

```
ENGINE M204IN SET MISSING {ON|OFF}
```

where:

ON

Uses the NODATA value to display MODEL 204 null data.

OFF

Is the default. Null values are not represented on reports.

Note:

- The default setting can affect the results of server DML SUM or COUNT aggregate operations. Null values may be counted or averaged in as existing values.
- You must issue the M204IN SET MISSING ON command in order to specify screening conditions (IF or WHERE MISSING tests) for null values.

Locking Records to Ensure Accurate Data Reports

When retrieving records, the MODEL 204 DBMS usually holds locks on the appropriate records to ensure accurate data for reports. The locks prevent other users from updating the target records while MODEL 204 constructs answer sets. The adapter generates standard IFFIND calls that lock the found set in SHR mode.

For certain circumstances, you can issue the adapter SET READWOL (read without locks) command to instruct the adapter to issue an IFFWOL (find without locks) HLI call. Use the following syntax to turn the setting ON:

```
ENGINE M204IN SET READWOL ON
```

For information about the design and performance considerations involved in deciding when to use the MODEL 204 IFFWOL command instead of the IFFIND command, consult the *MODEL 204 Host Language Adapter Programming Guide*.

Controlling Thread Management

The adapter M204IN SET SINGLETHREAD command enables users to control MODEL 204 thread management during record retrieval. A thread is a communications path or link between the adapter and the MODEL 204 DBMS. Adapter requests (generated HLI calls) are transmitted using threads. Threads are de-allocated after each adapter request is processed.

There are two types (or modes) of thread management:

- **Multiple.** The adapter creates as many threads as it needs whenever they are needed to access files.
- **Single.** The adapter creates one thread for all file access; this restricts processing to one request.

Syntax

How to Change the Mode of Thread Management

To change the mode, issue the following from the command level

```
ENGINE M204IN SET SINGLETHREAD {ON|OFF}
```

where:

ON

Forces the adapter to use a single thread for all file access.

OFF

Is the default. Allows the adapter to use multiple threads.

For example, if a request joins 10 files and the SINGLETHREAD setting is ON, only one thread is used to access all of the files. If the setting is OFF, up to 10 threads may be used. For performance reasons, the OFF setting is recommended.

Displaying Adapter Defaults and Current Settings

The adapter M204IN SET ? query command displays adapter defaults and current settings. Issue the following:

```
ENGINE M204IN SET ?
```

Adapter Tracing for MODEL 204

The Data Adapter for MODEL 204 Tracing Facility is activated from the Web Console main page.

Procedure How to Activate Adapter Tracing

Select *Diagnostics*, then *Traces*, and click *Enable Traces*.

The Default traces include the Data Adapter for MODEL 204 tracing information.

CHAPTER 22

Getting Started in MQSeries

Topics:

- Preparing the MQSeries Environment
- Configuring the Data Adapter for MQSeries
- Managing MQSeries Metadata

The Data Adapter for MQSeries allows applications to access MQSeries data sources. The adapter converts application requests into native MQSeries statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

Preparing the MQSeries Environment

The Data Adapter for MQSeries does not require any environment variables be set up.

The following variables appear on the console, but are for internal use.

- MQ_HOME
- MQSI_HOME

Configuring the Data Adapter for MQSeries

In the Data Adapter for MQSeries configuration screen, click Configure to add the Data Adapter for MQSeries.

Declaring Connection Attributes

Because a queue in MQSeries can be viewed as a sequential file, the server uses the logical layer of a flat file interface. In order to distinguish between sequential file storage types and MQSeries storage types, the FILEDEF command must be issued prior to any read/write request against MQSeries. FILEDEF is used to provide definitions for the queue manager, the queue name, and the optional message ID to the MQI layer.

Syntax How to Issue the FILEDEF Command

```
FILEDEF master QMAN queuemgr QUEUE queuename [MSGID msg_id]
```

where:

master

Is the Master File name used to provide the dynamic definitions of the queue manager, the queue, and the optional message ID.

queuemgr

Is the name of the queue manager.

queuename

Is the name of the queue.

msg_id

Is the optional parameter used when producing messages in a queue that are defined outside of the program assigned message ID.

Managing MQSeries Metadata

This topic describes how to use CREATE SYNONYM for MQSeries data sources. It also describes MQSeries data type support.

Creating Synonyms

Synonyms define unique names (or aliases) for each MQSeries data structure that is accessible from a server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while enabling client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

For details on how to create a synonym through the Web Console, see the Configuration and Operation manual for your platform.

Syntax

How to Create a Synonym Manually for Server Family Messages

```
CREATE SYNONYM appname/synonym FOR qmanager.queue.msgid DBMS MQS
```

where:

appname

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, then this application name must not be used.

synonym

Is an alias for the data source (maximum 64 characters for UNIX and Windows NT Server platforms).

qmanager.queue.msgid

Is the fully qualified name of the queue manager, the message queue, and the message ID if used.

MQS

Is the MQS value for the DBMS parameter that CREATE SYNONYM requires.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note:

- CREATE SYNONYM can span more than one line, however a single element cannot span more than one line.
- CREATE SYNONYM creates a Master File. An Access File is not created when creating a synonym from an MQSeries queue.

Example Retrieving Foreign Messages From the Queue

The Master File must be manually created. For example:

```
FILE      =MQS1,SUFFIX= MQS,$
DATASET=edarisc3.MQHOLD1.41423, $
SEGNAME=ROOT, SEGTYPE=S0,$
FIELD    =MESSAGE,      ALIAS=AA,  USAGE=A720,  A720  ,$
GROUP    =SPECIALS,     ALIAS=   ,  USAGE=A100,  A100  ,$
FIELD    =MSGID,        ALIAS=   ,  USAGE=A48,   A48   ,$
FIELD    =CORID,        ALIAS=   ,  USAGE=A48,   A48   ,$
FIELD    =TIMEOUT,      ALIAS=0   ,  USAGE=I4,   I4    ,$
```

Data Type Support

Data types in foreign messages are generated by application programs writing to the queues. If a Cobol copy book describing data layout exists, the metadata type could be generated by Cobol FD Translator.

The following chart provides information about the default mapping of MQSeries data types to server data types:

COBOL Format	Data Type	
	Actual	Usage
PICTURE X(n)	An	An
PICTURE 9(n)	Zn (packed option)	Pn (packed option)
PICTURE 9(n)	An (packed option)	Pn (packed option)
PICTURE S9(n)	Zn	Pn+1
PICTURE 9(n)V9(m)	Zn+m	Pn+m+1.m
PICTURE S9(n)V9(m)	Zn+m	Pn+m+2.m
PICTURE 9(n) COMP (1 <= n <= 4)	I2	I9
PICTURE 9(n) COMP (5 <= n <= 9)	I4	I9
PICTURE 9(n) COMP (n > 9)	A8	A8
COMP-1	F4	F8
COMP-2	D8	D15
PICTURE 9(n) COMP-3	P(n+2/2)	Pn

COBOL Format	Data Type	
	Actual	Usage
PICTURES 9(n) COMP-3	P(n+2/2)	Pn+1
PICTURE 9(n) V9(m) COMP-3	P(n+m+2/2)	Pn+m+1.m
PICTURE S9(n) V9(m) COMP-3	P(n+m+2/2)	Pn+m+2.m

The format conversions are subject to the following limitations:

- Alphanumeric fields are limited to 256 characters. Elementary fields that exceed 256 characters are truncated to 256 characters and have filler fields inserted to provide for the total length. Group fields that exceed 256 characters are commented.
- Unsigned zoned fields that exclude a decimal point are described with a USAGE of packed (P) or alphanumeric (A), depending on the value entered for the Zoned Numeric Field Usage option on the Translator menu.
- COMP-4 binary fields are described the same as COMP fields.
- Binary fields that exceed 9 digits in the picture clause are described with ACTUAL format A8 and USAGE format A8.

The maximum length of the packed USAGE format depends on the software release that uses the generated Master File. For releases that support long packed fields (16 or more digits), the maximum ACTUAL length is 16 bytes and the maximum USAGE length is 31 digits (or 32 characters including a decimal point and sign). For earlier releases, the maximum ACTUAL length is 8 bytes and the maximum USAGE length is 15 total digits, including decimal and sign. The number of decimal places is limited to one fewer than the total USAGE length. In general, the USAGE length of packed fields is calculated as the sum of the digits to the left of the decimal (n), the number of decimal positions (m), one for the leading minus sign if present (S), and one for the decimal (V) if present.

The ACTUAL and USAGE formats for GROUP fields are always alphanumeric (A). The ACTUAL length is the sum of the ACTUAL lengths of its components. The USAGE length is the sum of the following:

- The USAGE lengths of all the alphanumeric (A) components.
- 16 for each long packed (P) component.
- 8 for each short packed (P) and double precision (D) component.
- 4 for each integer (I) and floating point (F) component.

CHAPTER 23

Getting Started in Digital Standard MUMPS

Topics:

- Preparing the MUMPS Environment
- Configuring the Data Adapter for MUMPS
- Managing MUMPS Metadata

The Data Adapter for MUMPS allows applications to access MUMPS data sources. The adapter converts application requests into native MUMPS statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

Preparing the MUMPS Environment

To prepare the MUMPS environment on OpenVMS, the system administrator must set the following logicals:

```
DSM$DEFAULT_DSMAXP028 =  
/SOURCE_BUFFER_SIZE = 65535/SYMBOL_TABLE_SIZE=300000  
DSM$ENVIRONMENT      = DSMAXP028  
DSM$ID_DSMAXP028    = 00001
```

Configuring the Data Adapter for MUMPS

Configure the adapter using the Web Console Adapter Add screen and click *Configure*.

Syntax How to Configure the DSM MUMPS Database

First, you must configure the DSM MUMPS database that the server will be accessing.

You can do this with the following command, issued from the system prompt of the user ID that starts the server.

```
DEFINE DSM$DEFAULT_DSMAXP028/VOL=XXX/UCI=YYY
```

where:

XXX

Is the volume set group of DSM.

YYY

Is the user class definition.

Declaring Connection Attributes

Recovery unit mode can be switched to "ON" for the DSM database. If the database is in recovery mode, all commands must be issued inside the recovery unit. To set the adapter in this mode use, the DSM SET command:

```
ENGINE DSM SET RECOVERY_MODE {ON|OFF}
```

Authenticating a User

In any computer system, it is important that data be secured from unauthorized access. Both MUMPS and the server provide security mechanisms to ensure that users have access to only those objects for which they have authorization.

The system administrator must grant you access to the DSM environment.

Managing MUMPS Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the MUMPS data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each MUMPS table or view that is accessible from a server. Synonyms are useful because they hide the location and identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while enabling client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

Syntax How to Create a Synonym Manually

```
CREATE SYNONYM appname/synonym FOR global_name DBMS DSM DATABASE vol AT
uci
END
```

where:

appname

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, then this application name must not be used.

synonym

Is an alias for the data source (maximum 64 characters).

global_name

Is the name of the MUMPS global data structure.

vol

Is the volume set name where the global data structure exists.

uci

Is the user class identifier (uci) name of where the global exists.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note:

- CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.
- CREATE SYNONYM creates a Master File and Access File which represents the server metadata.

Example Using CREATE SYNONYM

Use the following syntax to create a synonym for the variable "Clinic."

```
CREATE SYNONYM Clinic FOR Clinic DBMS DSM DATABASE RAB AT MUA
Global Variable CLINIC:
^CLINIC("BROOKDALE CLINIC", "231-28-8833", "031291")
DR01,DG05,COR3XD063,45.00
^CLINIC("COMMUNITY CLINIC", "665-36-5843", "040191")
DR22,DG13,PSU1XW004,30.00
^CLINIC("COMMUNITY CLINIC", "834-48-8672", "041891")
DR67,DG12,CCL2XD021,25.00
^CLINIC("COMMUNITY CLINIC", "935-55-4254", "043091")
DR85,DG11,PEN3XW015,15.00
^CLINIC("GEORGETOWN CLINIC", "241-89-0001", "041891")
DR64,DG03,VAL2XD100,100.00
^CLINIC("GEORGETOWN CLINIC", "251-64-0011", "041791")
DR43,DG10,AMX3XD042,42.00
^CLINIC("GEORGETOWN CLINIC", "523-85-0007", "031791")
DR64,DG10,AMX2XD020,20.00
^CLINIC("GEORGETOWN CLINIC", "651-77-0009", "041791")
DR64,DG07,COD3XD063,63.00
^CLINIC("GEORGETOWN CLINIC", "656-64-0012", "041791")
DR43,DG07,TYL4XD050,25.00
^CLINIC("HILLSIDE CLINIC", "637-36-3615", "032291")
DR31,DG07,COD1XD021,60.00
^CLINIC("MAIN STREET CLINIC", "241-89-0001", "041891")
DR58,DG03,VAL2XD070,70.00
^CLINIC("MAIN STREET CLINIC", "251-64-0014", "041991")
DR73,DG11,PEN3XD042,84.00
```


Generated Clinic Master File:

```

FILE=CLINIC, SUFFIX=DSM, $
SEGMENT=GLOBAL, SEGTYPE=U, $
    FIELD=DUMMY, ALIAS=SUBSCRIPT, USAGE=A1, ACTUAL=A1, $
    FIELD=GLOBAL_VALUE, ALIAS=GLOBAL_VALUE, USAGE=A512, ACTUAL=A512, $
SEGMENT=SUB1, SEGTYPE=S0, $
    FIELD=SUB1, ALIAS=SUBSCRIPT, USAGE=A245, ACTUAL=A245, $
SEGMENT=SUB2, SEGTYPE=S0, $
    FIELD=SUB2, ALIAS=SUBSCRIPT, USAGE=A245, ACTUAL=A245, $
SEGMENT=SUB3, SEGTYPE=S0, $
    FIELD=SUB3, ALIAS=SUBSCRIPT, USAGE=A245, ACTUAL=A245, $
    FIELD=VAL3, ALIAS=VAL3, USAGE=A512, ACTUAL=A512, $

```

Generated Clinic Access File:

```

UCI=MUA, VOLUMESET=RAB, GLOBAL=CLINIC, CARDINALITY=6, WRITE=NO, $
SEGNAM=GLOBAL, DELIMITER=, $
SEGNAM=SUB3, DELIMITER=, $

```

Unlike other databases, MUMPS does not provide column names. Therefore, the API does not return column names. All columns are assigned names prefixed with SUB and numbered sequentially (SUB1, SUB2, ..., SUBn). You can change the field names generated by CREATE SYNONYM to make them more descriptive. Data value fields are placed in the segment associated with the subscript. Data values might contain a sequence of variable length fields delimited by special characters. These delimiting characters are described in the Access file and are associated with subscript segments. Redefinition of the data can be done to further facilitate reporting. This is also a manual update of the synonym. In this example, the previous master file synonym CLINIC, field VAL3 is redefined:

```

FILE=CLINIC, SUFFIX=DSM, $
SEGMENT=GLOBAL, SEGTYPE=U, $
    FIELD=DUMMY, ALIAS=SUBSCRIPT, USAGE=A1, ACTUAL=A1, $
    FIELD=GLOBAL_VALUE, ALIAS=GLOBAL_VALUE, USAGE=A512, ACTUAL=A512, $
SEGMENT=SUB1, SEGTYPE=S0, $
    FIELD=SUB1, ALIAS=SUBSCRIPT, USAGE=A245, ACTUAL=A245, $
SEGMENT=SUB2, SEGTYPE=S0, $
    FIELD=SUB2, ALIAS=SUBSCRIPT, USAGE=A245, ACTUAL=A245, $
SEGMENT=SUB3, SEGTYPE=S0, $
    FIELD=SUB3, ALIAS=SUBSCRIPT, USAGE=A245, ACTUAL=A245, $
    FIELD=VAL3, ALIAS=VAL3, USAGE=A512, ACTUAL=A512, $
SEGMENT=SUB3RED, SEGTYPE=S0, PARENT=SUB3, POSITION=VAL3, $
    FIELD=VAL3A, ALIAS=VAL3A, USAGE=A4, ACTUAL=A4, $
    FIELD=VAL3B, ALIAS=VAL3B, USAGE=A4, ACTUAL=A4, $
    FIELD=VAL3C, ALIAS=VAL3C, USAGE=A9, ACTUAL=A9, $
    FIELD=VAL3D, ALIAS=VAL3B, USAGE=P6.2, ACTUAL=A6, $

```

Data Type Support

All data in DSM MUMPS is represented as alphanumeric. The format of the data can be changed manually in the USAGE attribute of the Master File. For example, see the VAL3D field in the above example.

CHAPTER 24

Getting Started in MySQL

Topics:

- Preparing the MySQL Environment
- Configuring the Data Adapter for MySQL
- Managing MySQL Metadata
- Customizing the MySQL Adapter Environment

The Data Adapter for MySQL allows applications to access MySQL data sources. The adapter converts application requests into native MySQL statements and returns optimized answer sets to the requesting application.

Preparing the MySQL Environment

For the MySQL Data Adapter, no environment variables need to be set prior to starting the server.

Configuring the Data Adapter for MySQL

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

The SET CONNECTION_ATTRIBUTES command allows you to declare a connection to one MySQL database server and to supply authentication attributes necessary to connect to the server.

You can declare connections to more than one MySQL database server by issuing multiple SET CONNECTION_ATTRIBUTES commands. The actual connection takes place when the first query referencing that connection is issued (see *Overriding the Default Connection* on page 24-4). You can include SET CONNECTION_ATTRIBUTES commands in an RPC or a server profile. The profile can be encrypted.

If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The first SET CONNECTION_ATTRIBUTES command sets the default MySQL database server to be used.
- If more than one SET CONNECTION_ATTRIBUTES command declares the same MySQL database server, the authentication information is taken from the last SET CONNECTION_ATTRIBUTES command.

Connecting to an MySQL Database Server

Using the standard rules for deploying MyODBC, the server supports connections to:

- Local MySQL database servers.
- Remote MySQL database servers. To connect to a remote MySQL database server, odbc.ini file on the source machine must contain an entry for the MySQL data source name of the target machine and the listening process must be running on the target machine.

Authenticating a User

There are two methods by which a user can be authenticated when connecting to an MySQL database server:

- **Explicit.** User IDs and passwords are explicitly stated in SET CONNECTION_ATTRIBUTES commands. You can include these commands in the server global profile, edasprof.prf, for all users.
- **Database or Password Passthru.** User ID and password received from the client application are passed to the MySQL database server for authentication.

When a client connects to the server, the user ID and password are passed to MySQL for authentication and are not authenticated by the server. To implement this type of authentication, start the server with security turned off. The server allows the client connection, and then stores an encrypted form of the client's connection message to be used for connection to an MySQL database server at anytime during the lifetime of the server agent.

Syntax How to Declare Connection Attributes

For explicit authentication:

```
ENGINE [MYSQL] SET CONNECTION_ATTRIBUTES
[connection_name]/userid,password
```

For Password Passthru authentication:

```
ENGINE [MYSQL] SET CONNECTION_ATTRIBUTES connection_name/
```

where:

MYSQL

Indicates the MySQL Data Adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection_name

MySQL data source name as defined in the odbc.ini file

userid

Is the primary authorization ID by which you are known to MySQL.

password

Is the password associated with the primary authorization ID.

Example Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to the MySQL database server named TEST with an explicit user ID and password:

```
ENGINE MYSQL SET CONNECTION_ATTRIBUTES TEST/USERA,PWDA
```

The following SET CONNECTION_ATTRIBUTES command connects to the MySQL database server named TEST using Password Passthru authentication:

```
ENGINE MYSQL SET CONNECTION_ATTRIBUTES TEST/
```

Adapter Configuration

The Web Console MySQL Adapter configuration screen displays the following fields:

Field	Description
Datasource	Is a valid MySQL ODBC data source name.
Security	There are two methods by which a user can be authenticated when connecting to an MySQL database server: Explicit. The user ID and password are explicitly specified for each connection and passed to MySQL, at connection time, for authentication. Password Passthru. The user ID and password received from the client application are passed to MySQL, at connection time, for authentication. This option requires that the server be started with security off.
User	Is the username for the MySQL authenticated login.
Password	Is the password that identifies the entered username.

The values for these fields will be used to describe connection attributes for the MySQL server.

Overriding the Default Connection

Once all MySQL connections to be accessed have been declared using the SET CONNECTION_ATTRIBUTES command, there are two ways to select a specific MySQL connection from the list of declared connections:

- You can select a default connection using the SET DEFAULT_CONNECTION command. If you do not issue this command, the connection_name value specified in the *first* SET CONNECTION_ATTRIBUTES command is used.

- You can include the CONNECTION= attribute in the Access File of the table specified in the current SQL query. When you include a connection name in the CREATE SYNONYM command from the Web Console, the CONNECTION= attribute is automatically included in the Access File. This attribute supersedes the default connection.

Syntax **How to Select a Connection to Access**

`ENGINE [MYSQL] SET DEFAULT_CONNECTION [connection_name]`

where:

`MYSQL`

Indicates the MySQL Data Adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection_name`

Is the connection name specified in a previously issued SET CONNECTION_ATTRIBUTES command. If omitted, then the local database server will be set as the default. If this connection name has not been previously declared, a FOC1671 message is issued.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection_name specified in the last command will be the active connection_name.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, a FOC1671 message is issued.

Example **Selecting a Connection to Access**

The following SET DEFAULT_CONNECTION command selects the MySQL database server named TNSNAMEB as the default MySQL database server:

`ENGINE MYSQL SET DEFAULT_CONNECTION datasource_name`

Note: You must have previously issued a SET CONNECTION_ATTRIBUTES command for the datasource_name

Controlling Connection Scope

This topic explains how to set the scope of logical units of work using data adapters. This is accomplished by the SET AUTODISCONNECT command.

A connection occurs at the first interaction with the declared database server.

Syntax How to Control the Connection Scope

```
ENGINE [MYSQL] SET AUTODISCONNECT ON {FIN|COMMAND}
```

where:

[MYSQL](#)

Indicates the MySQL Data Adapter. You can omit this value if you previously issued the SET SQLENGINE command.

[FIN](#)

Disconnects automatically only after the session has been terminated. This value is the default.

[COMMAND](#)

Disconnects automatically after each request. Depending on how often the event occurs, the AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server; it is related to the operating system and the data source.

Managing MySQL Metadata

This topic describes how to use CREATE SYNONYM for MySQL data sources. It also describes MySQL data type support.

Creating Synonyms

Synonyms define unique names (or aliases) for each MySQL table or view that is accessible from the server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

Procedure How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Editing Metadata page opens.

To create a synonym, you must have configured the adapter. For information on how to configure an adapter, see *Adapter Configuration* on page 24-4.

2. Expand the Add folder, expand the adapter folder, and then click a connection. The right pane displays table and view selection options:

Select All Tables/Views. Select this radio button to create synonyms for all tables and views. This value is the default.

Filter by Name, Owner and Table Type. Select this radio button to filter the tables or views for which to create synonyms.

Selecting this option adds the following:

Owner. Enter a string for filtering the owners' IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owners' IDs begin with the letters ABC; %ABC to select tables or views whose owners' IDs end with the letters ABC; %ABC% to select tables or views whose owners' IDs contain the letters ABC at the beginning, middle, or end.

Table Name. Enter a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

Table Type. Select one of the following options: Tables (this is the default), Views, Tables/Views.

3. Click *Select Tables*. All tables that meet the specified criteria are displayed:
4. If you have tables with identical table names, assign a prefix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave field blank.

5. From the SET CONVERSION LONGCHAR drop-down list, select: *Text*, *Alpha*, or *BLOB*. The default value is *Text*.
6. From the Select Application Directory drop-down list, select a directory. The default value is baseapp.

- Complete your table or view selection:

To select all tables or views in the list, click *All*.

To select specific tables or views, click the corresponding checkboxes.

- The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.
- Click *Create Synonym*. Synonyms are created and added under the specified application directory.

Once a synonym is added, clicking the name of the synonym in the navigation pane opens a menu with the following options:

Sample Data	Retrieves up to 20 rows from the associated table.
View Metadata	Displays a graphic representation of the synonym. Click on a row to view its description.
Edit Master File	Enables you to manually edit the synonym's Master File. Note: It is strongly recommended that you use the Refresh option, rather than manually editing the Master File, to update the synonym.
Edit Access File	Enables you to manually edit the synonym's Access File. Note: It is strongly recommended that you use the Refresh option, rather than manually editing the Access File, to update the synonym.
Refresh	Regenerates the synonym. Use this option if structural changes were made to the source table or view.
Drop	Deletes the synonym.
Copy to	Copies the synonym to another application directory. Click the target directory from the resulting list.
Move to	Movies the synonym to another application directory. Click the target directory from the resulting list.

Syntax **How to Create a Synonym Manually**

```
CREATE SYNONYM appname/synonym FOR table_view DBMS MYSQL [AT
connection_name] [NOCOLS]
[AT ' ']
```

END

where:

appname

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, then this application name must not be used.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. See the MySQL documentation for specific naming conventions.

MYSQL

Indicates the MySQL Data Adapter.

AT *connection_name*

Is the *connection_name* as previously specified in a SET CONNECTION_ATTRIBUTES command. When the server creates the synonym, this >*connection_name* becomes the value for CONNECTION= in the Access File.

AT ' '

Includes CONNECTION=' ', \$ in the Access File. A query against this synonym will access the local MySQL database server.

NOCOLS

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

CREATE SYNONYM creates a Master File and Access File which represents the server metadata.

Example CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.NF29004 DBMS MYSQL AT TEST NOCOLS
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION ,SUFFIX=MYSQL,$
SEGNAME=SEG1_4 ,SEGTYPE=S0 ,$
FIELD=DIVISION4 ,DIVISION4 ,I9 ,I4 ,MISSING=OFF ,$
FIELD=DIVISION_NA4,DIVISION_NA4 ,A25 ,A25 ,MISSING=ON ,$
FIELD=DIVISION_HE4,DIVISION_HE4 ,I9 ,I4 ,MISSING=ON ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4, TABLENAME=EDAQA.NF29004,
CONNECTION=TEST, KEYS=1, WRITE=YES, $
```

Reference Access File Keywords

Keyword	Description
TABLENAME	Identifies the MySQL tablename. The tablename may include owner- (schema-) name. For example, <code>TABLENAME=[owner.]tablename</code>
CONNECTION= <i>connection_name</i>	Indicates access to specified database server.
CONNECTION=' '	Indicates access to local database server.
Absence of CONNECTION=	Indicates access to default database server.

Data Type Support

The following chart provides information about the default mapping of MySQL data types to server data types:

MySQL Data Type	Data Type		Remarks
	Usage	Actual	
TINYINT[(m)] m in signed range (-128...127), unsigned range (0...255)	I6	I4	BIT and BOOL are synonyms for TINYINT(1)
SMALLINT[(m)] m in signed range (-32768...32767) unsigned range (0...65535)	I6	I4	
MEDIUMINT[(m)] m in signed range (-8388608...8388607) unsigned range (0...16777215)	I11	I4	
INT[(m)] m in signed range (-2147483648...2147483647) unsigned range (0...4294967295)	I11	I4	
BIGINT[(m)] signed range is (-9223372036854775808...9223372036854775807) unsigned range is (0...18446744073709551615)	P20	P10	
FLOAT(p) p in (1...24)	D20.2	D8	
DOUBLE(p) p in (25...53)	D20.2	D8	REAL is a synonym for DOUBLE
DECIMAL [(m[d])] m (25...53) DEC (10,0)- default value	P11	P8	NUMERIC is a synonym for DECIMAL
DATE	YYMD	DATE	
DATETIME	HYYMDS	HYYMDS	

MySQL Data Type	Data Type		Remarks
	Usage	Actual	
TIMESTAMP	HYYMDS	HYYMDS	
TIME	HHIS	HHIS	
YEAR	I6	I4	
CHAR(<i>m</i>) NATIONAL CHAR (<i>m</i>)	<i>An</i>	<i>An</i>	<i>m</i> is an integer between 0 and 255 <i>n</i> is an integer between 1 and 255
VARCHAR (<i>m</i>) NATIONAL VARCHAR (<i>m</i>)	<i>An</i>	<i>An</i>	<i>m</i> is an integer between 1 and 255 <i>n</i> is an integer between 1 and 255
TINYBLOB TINYTEXT	A32767	A32767	
BLOB TEXT	A32767	A32767	
MEDIUMBLOB MEDIUMTEXT	A32767	A32767	
LOBLOB LONGTEXT	A32767	A32767	
ENUM('value1', 'value2',.....) max of 65535 values	Not Supported	Not Supported	
SET ('value1', 'value2',.....) max of 64 members	Not Supported	Not Supported	

Precision and Scale

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Syntax

How to Override Default Precision and Scale

```
ENGINE [MYSQL] SET CONVERSION RESET
ENGINE [MYSQL] SET CONVERSION format RESET
ENGINE [MYSQL] SET CONVERSION format [PRECISION nn [mm]]
ENGINE [MYSQL] SET CONVERSION format [PRECISION MAX]
```

where:

MYSQL

Indicates the MySQL Data Adapter. You can omit this value if you previously issued the SET SQLENGINE command.

RESET

Returns any previously specified precision and scale values to the data adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

format

Is any valid format supported by the data source. Possible values are:

INTEGER

Indicates that the command applies only to INTEGER and SMALLINT columns.

DECIMAL

Indicates that the command applies only to DECIMAL columns.

FLOAT

Indicates that the command applies only to double precision floating point columns.

nn

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

mm

Is the scale. This is valid with DECIMAL and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. If the scale is not required, you must set mm to 0 (zero).

MAX

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example **Setting the Precision and Scale Attributes**

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE MYSQL SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE MYSQL SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE MYSQL SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE MYSQL SET CONVERSION RESET
```


CLOB Activation

The default mapping for MySQL data types CHAR, VARCHAR, VARCHAR2, and RAW is the data type ALPHA. The ALPHA data type maximum supported length is 4096 characters for TABLE/MODIFY and 32768 characters for API applications. It is now possible to perform SELECT, INSERT, and UPDATE on columns longer than 256 with these data types without using the server data type CLOB.

Syntax **How to Set Server CLOB Activation**

To activate the support for the server data type CLOB, you must issue the following command in one of the supported server profiles:

```
ENGINE [MYSQL] SET CONVERSION LONGCHAR TEXT
```

where:

MYSQL

Indicates the MySQL Data Adapter. You can omit this value if you previously issued the SET SQLENGINE command.

TEXT

Activates support of long character columns as Character Large Objects. The default is ALPHA.

Customizing the MySQL Adapter Environment

This topic describes operational and performance adapter settings including optimization and bulk operations.

PASSRECS

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru UPDATE or DELETE command.

Syntax

How to Set PASSRECS

```
ENGINE [MYSQL] SET PASSRECS {ON|OFF}
```

where:

MYSQL

Indicates the MySQL Data Adapter. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru UPDATE or DELETE command. This value is the default.

OFF

Provides no information after the successful execution of an SQL Passthru UPDATE or DELETE command.

In addition, the data adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

CHAPTER 25

Getting Started in Nucleus

Topics:

- Preparing the Nucleus Environment
- Configuring the Data Adapter for Nucleus
- Managing Nucleus Metadata
- Customizing the Nucleus Environment

The Data Adapter for Nucleus allows applications to access Nucleus data sources. The adapter converts application requests into native Nucleus statements and returns optimized answer sets to the requesting application.

Preparing the Nucleus Environment

In order to use the Nucleus Data Adapter, the Nucleus database and ODBC must be installed and configured and the path to the Nucleus ODBC libraries directory must be added to SYSTEM LIBRARY PATH. See the SandTechnology Nucleus[®] documentation for more information about requirements for ODBC installation and configuration.

Procedure How to Set Up the Environment on Windows NT/2000

On Windows NT/2000, the Nucleus environment is set up during the installation of Nucleus.

Procedure How to Set Up the Environment on UNIX

You can access Nucleus using the NUCINI environment variable and one of the following: \$HOME/.odbc.ini file or ODBCINI environment variable.

Point the NUCINI variable to the directory where SandTechnology ODBC[®] is installed. For example:

```
NUCINI=/usr/nucleus/odbc
export NUCINI
```

The ODBCINI variable holds the absolute path to the *.ini file, which describes the Nucleus data sources. For example:

```
ODBCINI=/usr/ibiuser/odbc/nucodbc.ini
export ODBCINI
```

Configuring the Data Adapter for Nucleus

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to an Nucleus database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Manually add the command in the global server profile (edasprof.prf) or in a user profile (user.prf).
- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the global server profile. For more information, see *Declaring Nucleus Connection Attributes From the Web Console* on page 25-4.

You can declare connections to more than one Nucleus database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to Nucleus Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Syntax **How to Declare Connection Attributes Manually**

```
ENGINE [SQLNUC] SET CONNECTION_ATTRIBUTES [connection]/user_ID,password
```

where:

SQLNUC

Indicates the Data Adapter for Nucleus. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the name of the Nucleus DSN (Data Source Name) you wish to access. It must match an entry in the .odbc.ini.

user_ID

Is the primary authorization ID by which you are known to Nucleus.

password

Is the password associated with the primary authorization ID.

Example **Declaring Connection Attributes**

The following SET CONNECTION_ATTRIBUTES command connects to the Nucleus database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLNUC SET CONNECTION_ATTRIBUTES SAMPLESERVER/MYUSER,PASS
```

Reference **Declaring Nucleus Connection Attributes From the Web Console**

Attribute	Description
Data source	Enter the valid Nucleus DSN (data source name). There is no default DSN; a value must be entered.
User	Enter the valid Nucleus user name.
Password	Enter the password that identifies the entered user name.

Overriding the Default Connection

Once connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax How to Change the Default Connection

```
ENGINE [SQLNUC] SET DEFAULT_CONNECTION [connection]
```

where:

SQLNUC

Indicates the Data Adapter for Nucleus. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the service name defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following error message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following error message is issued: FOC1671, Command out of sequence.

Example Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the Nucleus database server named SAMPLENAME as the default Nucleus database server:

```
ENGINE SQLNUC SET DEFAULT_CONNECTION SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax **How to Control the Connection Scope**

```
ENGINE [SQLNUC] SET AUTODISCONNECT ON {FIN|COMMIT|COMMAND}
```

where:

SQLNUC

Indicates the Data Adapter for Nucleus. You can omit this value if you previously issued the SET SQLENGINE command.

FIN

Disconnects automatically only after the session has been terminated. This value is the default.

COMMIT

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

COMMAND

Disconnects automatically after each request. Depending on how often the event occurs, the AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server; it is related to the operating system and the data source.

Managing Nucleus Metadata

When the server accesses a data source, it needs to know how to interpret the data that it finds. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Nucleus data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Nucleus table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click Metadata. The Editing Metadata page opens.

To create a synonym, you must have configured the adapter. See *Configuring the Data Adapter for Nucleus* on page 25-3 for more information.

2. Expand the *Add* folder, expand the adapter folder, and then click a connection. The right pane displays table and view selection options:

Select All Tables/Views. Select this radio button to create synonyms for all tables and views. This value is the default.

Filter by Name, Owner and Table Type. Select this radio button to filter the tables or views for which to create synonyms.

Selecting this option adds the following:

The screenshot shows a dark blue panel with the following elements:

- Two radio buttons:
 - Select all Tables/Views
 - Filter by Name, Owner and Table Type
- Three input fields:
 - Owner: [] - Sample: abc%
 - Table name: [NF%] - Sample: abc%
 - Table type: TABLE VIEW TABLE/VIEW
- A button at the bottom labeled "Select Tables".

Owner. Enter a string for filtering the owners' IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owners' IDs begin with the letters ABC; %ABC to select tables or views whose owners' IDs end with the letters ABC; %ABC% to select tables or views whose owners' IDs contain the letters ABC at the beginning, middle, or end.

Table Name. Enter a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

Table Type. Select one of the following options: Tables (this is the default), Views, Tables/Views.

3. Click *Select Tables*. All tables that meet the specified criteria are displayed:

Prefix: SET CONVERSION LONGCHAR:

Select Application Directory:

All Select

Check	Default Synonym Name	Owner/Schema	Table Name	Type
<input type="checkbox"/>	<input type="text" value="NF29001"/>	EDAQA	NF29001	TABLE
<input type="checkbox"/>	<input type="text" value="NF29002"/>	EDAQA	NF29002	TABLE
<input type="checkbox"/>	<input type="text" value="NF29003"/>	EDAQA	NF29003	TABLE
<input type="checkbox"/>	<input type="text" value="NF29004"/>	EDAQA	NF29004	TABLE
<input type="checkbox"/>	<input type="text" value="NF29005"/>	EDAQA	NF29005	TABLE
<input type="checkbox"/>	<input type="text" value="NF29006"/>	EDAQA	NF29006	TABLE
<input type="checkbox"/>	<input type="text" value="NF29007"/>	EDAQA	NF29007	TABLE
<input type="checkbox"/>	<input type="text" value="NF29008"/>	EDAQA	NF29008	TABLE
<input type="checkbox"/>	<input type="text" value="NF29009"/>	EDAQA	NF29009	TABLE
<input type="checkbox"/>	<input type="text" value="NF29012"/>	EDAQA	NF29012	TABLE
<input type="checkbox"/>	<input type="text" value="NF29013"/>	EDAQA	NF29013	TABLE

4. If you have tables with identical table names, assign a prefix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave field blank.

5. From the Select Application Directory drop-down list, select a directory. The default value is baseapp.

6. Complete your table or view selection:
To select all tables or views in the list, click **All**.
To select specific tables or views, click the corresponding check boxes.
7. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.
8. Click *Create Synonym*. Synonyms are created and added under the specified application directory.

Reference **Managing Synonyms**

In the navigation pane, click the name of the synonym to access the following options:

Sample Data	Retrieves up to 20 rows from the associated data source.
Properties	Displays a graphic representation of the synonym and enables you to edit its metadata.
Edit Master File	Enables you to manually edit the synonym's Master File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Master File, to update the synonym.
Edit Access File	Enables you to manually edit the synonym's Access File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Access File, to update the synonym.
Refresh	Regenerates the synonym. Use this option if structural changes were made to the data source.
Drop	Deletes the synonym.
Copy to	Copies the synonym to another application directory. Click the target directory from the resulting list.
Move to	Moves the synonym to another application directory. Click the target directory from the resulting list.

Syntax **How to Create a Synonym Manually**

```
CREATE SYNONYM app/synonym FOR datasource DBMS SQLNUC  
[AT connection][NOCOLS]  
END
```

where:

app

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, then this application name must not be used.

synonym

Is an alias for the data source (maximum 64 characters).

datasource

Is the fully qualified name for the physical data structure, such as [database.]schema.tablename in Nucleus for example. See the Nucleus documentation for specific naming conventions.

AT *connection*

Is the name of the connection as previously specified in a SET CONNECTION_ATTRIBUTES command.

When the server creates the synonym, this connection name becomes the value for CONNECTION= in the Access File. For Nucleus, this is the Data Source Name (DSN) as it is defined in nucleus.ini and in .odbc.ini.

If, after creating the synonyms with this option, the DBMS server's environment changes (for example, if tables are moved to a different DBMS server), you must either create new synonyms or edit existing ones.

If this parameter is omitted, the server uses DBMS-specific settings in its environment.

SQLNUC

Indicates the Data Adapter for Nucleus.

NOCOLS

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.NF29004 DBMS SQLNUC AT DSN_A
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION ,SUFFIX=SQLNUC,$
SEGNAME=SEG1_4 ,SEGTYPE=S0 ,$
FIELD=DIVISION4 ,DIVISION4 ,I9 ,I4 ,MISSING=OFF ,$
FIELD=DIVISION_NA4,DIVISION_NA4 ,A25 ,A25 ,MISSING=ON ,$
FIELD=DIVISION_HE4,DIVISION_HE4 ,I9 ,I4 ,MISSING=ON ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4 ,TABLENAME=EDAQA.NF29004 ,
CONNECTION=DSN_A ,KEYS=1 ,WRITE=YES , $
```

Reference Access File Keywords

Keyword	Description
TABLENAME	Identifies the Nucleus table. The value assigned to this attribute can include the name of the owner (also known as schema) as follows: <code>TABLENAME=[owner.]table</code>
CONNECTION	Indicates a previously declared connection. The syntax is: <code>CONNECTION=connection</code> CONNECTION=' ' indicates access to the local Nucleus database server. Absence of the CONNECTION attribute indicates access to the default database server.

Data Type Support

The following chart provides information about the default mapping of Nucleus data types to server data types:

Nucleus Data Type	Data Type		Remarks
	Usage	Actual	
CHAR (n)	An	An	n ranges in 1...2040
VARCHAR (n)	An	An	n ranges in 1...2040
SMALLINT (-32768... 32767)	I6	I4	
INTEGER (-2 31... 2 31 -1)	I11	I4	$2^{31} = 2,147,483,648$.
NUMERIC, DECIMAL (p,s)	D20.2	D8	See <i>Changing the Precision and Scale of Numeric Columns</i> on page 25-13.
FLOAT	D20.2	D8	
REAL	D20.2	D8	
DOUBLE PRECISION	D20.2	D8	

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Syntax

How to Override Default Precision and Scale

```
ENGINE [SQLNUC] SET CONVERSION RESET
ENGINE [SQLNUC] SET CONVERSION format RESET
ENGINE [SQLNUC] SET CONVERSION format [PRECISION pp [ss]]
ENGINE [SQLNUC] SET CONVERSION format [PRECISION MAX]
```

where:

SQLNUC

Indicates the Data Adapter for Nucleus. You can omit this value if you previously issued the SET SQLENGINE command.

RESET

Returns any previously specified precision and scale values to the data adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

format

Is any valid format supported by the data source. Possible values are:

INTEGER which indicates that the command applies only to INT columns.

DECIMAL which indicates that the command applies only to DECIMAL columns.

FLOAT which indicates that the command applies only to double precision floating point columns.

pp

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

ss

Is the scale. This is valid with DECIMAL and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. If the scale is not required, you must set *ss* to 0 (zero).

MAX

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE SQLNUC SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLNUC SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE SQLNUC SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLNUC SET CONVERSION RESET
```

Customizing the Nucleus Environment

The Data Adapter for Nucleus provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Syntax **How to Obtain the Number of Rows Updated or Deleted**

```
ENGINE [SQLNUC] SET PASSRECS {ON|OFF}
```

where:

SQLNUC

Indicates the Data Adapter for Nucleus. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. This value is the default.

OFF

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

CHAPTER 26

Getting Started in ODBC

Topics:

- Preparing the ODBC Environment
- Configuring the Data Adapter for ODBC
- Managing ODBC Metadata
- Customizing the ODBC Environment

The Data Adapter for ODBC allows applications to access ODBC data sources. The adapter converts application requests into native ODBC statements and returns optimized answer sets to the requesting application.

Preparing the ODBC Environment

In order to use the ODBC Data Adapter, you must install the Microsoft 32-bit ODBC Driver Manager on the Windows NT/2000 systems. The ODBC Data Adapter accesses all ODBC drivers installed and defined in the 32-bit ODBC Driver Manager.

Configuring the Data Adapter for ODBC

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to an ODBC database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Manually add the command in the global server profile (edasprof.prf) or in a user profile (user.prf).
- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the global server profile. For more information, see *Declaring ODBC Connection Attributes From the Web Console*.

You can declare connections to more than one ODBC database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to ODBC Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Syntax How to Declare Connection Attributes Manually

```
ENGINE [SQLODBC] SET CONNECTION_ATTRIBUTES [data source]/user_ID,password
```

where:

SQLODBC

Indicates the Data Adapter for ODBC. You can omit this value if you previously issued the SET SQLENGINE command.

data source

Is the name of the ODBC data source you wish to access.

user_ID

Is the primary authorization ID by which you are known to ODBC.

password

Is the password associated with the primary authorization ID.

Example Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to the ODBC database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLODBC SET CONNECTION_ATTRIBUTES SAMPLESERVER/MYUSER,PASS
```

Reference Declaring ODBC Connection Attributes From the Web Console

Attribute	Description
Data source	Enter the data source name configured using the ODBC Driver Manager.
User	Enter the primary authorization ID by which you are known to ODBC data source.
Password	Enter the password associated with the primary authorization ID.

Overriding the Default Connection

Once connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax How to Change the Default Connection

```
ENGINE [SQLODBC] SET DEFAULT_CONNECTION [connection]
```

where:

SQLODBC

Indicates the Data Adapter for ODBC. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the service name defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following error message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following error message is issued: FOC1671, Command out of sequence.

Example **Selecting the Default Connection**

The following SET DEFAULT_CONNECTION command selects the ODBC database server named SAMPLENAME as the default ODBC database server:

```
ENGINE SQLODBC SET DEFAULT_CONNECTION SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax **How to Control the Connection Scope**

```
ENGINE [SQLODBC] SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

SQLODBC

Indicates the Data Adapter for ODBC. You can omit this value if you previously issued the SET SQLENGINE command.

FIN

Disconnects automatically only after the session has been terminated. This value is the default.

COMMIT

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing ODBC Metadata

When the server accesses a data source, it needs to know how to interpret the data that it finds. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the ODBC data types.

Identifying the Data Adapter

The SUFFIX attribute in the Master File identifies the data adapter needed to interpret a request. Use the SUFFIX value SQLODBC to identify the ODBC Data Adapter.

Syntax How to Identify the ODBC Data Adapter

```
FILE[NAME]=file, SUFFIX=SQLODBC [,,$]
```

where:

file

Is the file name for the Master File. The file name without the .mas extension can consist of a maximum of eight alphanumeric characters. The file name should start with a letter and be representative of the table or view contents. The actual file must have a .mas extension, but the value for this attribute should not include the extension.

SQLODBC

Is the value for the ODBC Data Adapter.

Accessing Database Tables

If you choose to access a remote third-party table using ODBC, you must locally install the RDBMS' ODBC Driver.

The iWay Server can access third-party database tables across the network using the ODBC Data Adapter. You must provide a system data source name and possibly a user ID and/or password for the database tables you are accessing. You can define these parameters in either the server's global profile or in a user profile.

Creating Synonyms

Synonyms define unique names (or aliases) for each ODBC table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Editing Metadata page opens.

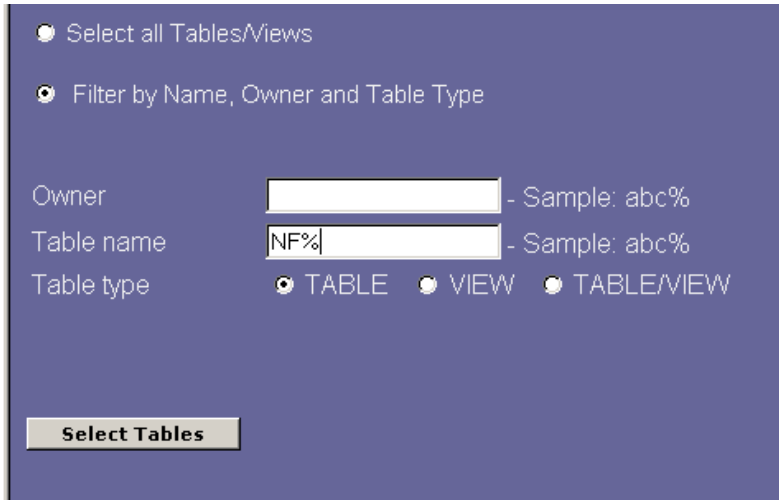
To create a synonym, you must have configured the adapter. See *Configuring the Data Adapter for ODBC* for more information.

2. Expand the *Add* folder, expand the adapter folder, and then click a connection. The right pane displays table and view selection options:

Select All Tables/Views. Select this radio button to create synonyms for all tables and views. This value is the default.

Filter by Name, Owner and Table Type. Select this radio button to filter the tables or views for which to create synonyms.

Selecting this option adds the following:



Select all Tables/Views

Filter by Name, Owner and Table Type

Owner - Sample: abc%

Table name - Sample: abc%

Table type TABLE VIEW TABLE/VIEW

Owner. Enter a string for filtering the owners' IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owners' IDs begin with the letters ABC; %ABC to select tables or views whose owners' IDs end with the letters ABC; %ABC% to select tables or views whose owners' IDs contain the letters ABC at the beginning, middle, or end.

Table Name. Enter a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

Table Type. Select one of the following options: Tables (this is the default), Views, Tables/Views.

3. Click *Select Tables*. All tables that meet the specified criteria are displayed:

Prefix: SET CONVERSION LONGCHAR: TEXT (for Web Focus) ▼

Select Application Directory: baseapp ▼

All Select

Check	Default Synonym Name	Owner/Schema	Table Name	Type
<input type="checkbox"/>	<input type="text" value="NF29001"/>	EDAQA	NF29001	TABLE
<input type="checkbox"/>	<input type="text" value="NF29002"/>	EDAQA	NF29002	TABLE
<input type="checkbox"/>	<input type="text" value="NF29003"/>	EDAQA	NF29003	TABLE
<input type="checkbox"/>	<input type="text" value="NF29004"/>	EDAQA	NF29004	TABLE
<input type="checkbox"/>	<input type="text" value="NF29005"/>	EDAQA	NF29005	TABLE
<input type="checkbox"/>	<input type="text" value="NF29006"/>	EDAQA	NF29006	TABLE
<input type="checkbox"/>	<input type="text" value="NF29007"/>	EDAQA	NF29007	TABLE
<input type="checkbox"/>	<input type="text" value="NF29008"/>	EDAQA	NF29008	TABLE
<input type="checkbox"/>	<input type="text" value="NF29009"/>	EDAQA	NF29009	TABLE
<input type="checkbox"/>	<input type="text" value="NF29012"/>	EDAQA	NF29012	TABLE
<input type="checkbox"/>	<input type="text" value="NF29013"/>	EDAQA	NF29013	TABLE

4. If you have tables with identical table names, assign a prefix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave field blank.

5. From the SET CONVERSION LONGCHAR drop-down list, select: Text, Alpha, or BLOB. The default value is Text.
6. From the Select Application Directory drop-down list, select a directory. The default value is baseapp.
7. Complete your table or view selection:

To select all tables or views in the list, click **All**.

To select specific tables or views, click the corresponding check boxes.

8. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.
9. Click *Create Synonym*. Synonyms are created and added under the specified application directory.

Reference **Managing Synonyms**

In the navigation pane, click the name of the synonym to access the following options:

Sample Data	Retrieves up to 20 rows from the associated data source.
Properties	Displays a graphic representation of the synonym and enables you to edit its metadata.
Edit Master File	Enables you to manually edit the synonym's Master File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Master File, to update the synonym.
Edit Access File	Enables you to manually edit the synonym's Access File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Access File, to update the synonym.
Refresh	Regenerates the synonym. Use this option if structural changes were made to the data source.
Drop	Deletes the synonym.
Copy to	Copies the synonym to another application directory. Click the target directory from the resulting list.
Move to	Moves the synonym to another application directory. Click the target directory from the resulting list.

Syntax **How to Create a Synonym Manually**

```
CREATE SYNONYM app/synonym FOR table_view DBMS SQLODBC [AT connection]
[NOCOLS]
END
```

where:

app

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, then this application name must not be used.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema) and the database link name as follows:

```
[owner.] table [@database link]
```

SQLODBC

Indicates the Data Adapter for ODBC.

AT *connection*

Is the DSN name configured using the 32-bit ODBC Driver Manager. When the server creates the synonym, this command becomes the value for SERVER= in the Access File. Use this option when dynamically switching server access from one DBMS server to another.

If, after creating the synonyms with this option, the DBMS server's environment changes (for example, if the mdb file or dsn is renamed), you must either create new synonyms or edit existing ones.

If this parameter is omitted, the server uses DBMS- specific settings in its environment.

NOCOLS

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example Using CREATE SYNONYM

The following example creates a synonym, named nf29004, for the ODBC table named NF29004.

```
CREATE SYNONYM nf29004 FOR EDAQA.NF29004 DBMS ODB AT DB1
```

The synonym creates the following Master File and Access File.

Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=ODB,$
SEGNAME=SEG1_4, SEGTYPE=S0,$
FIELD=DIVISION4, DIVISION4, I9, I4, MISSING=OFF,$
FIELD=DIVISION_NA4, DIVISION4, I9, I4, MISSING=OFF,$
FIELD=DIVISION_HE4, DIVISION4, I9, I4, MISSING=OFF,$
```

Access File nf29004.acx

```
SEGNAME=SEG1_4, TABLENAME=EDAQA.NF29004,
CONNECTION=DB1, KEYS=1, WRITE=YES, $
```

Reference Access File Keywords

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the ODBC table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: <code>TABLENAME=[owner.]table [@databaselink]</code>
CONNECTION	Indicates a previously declared connection. The syntax is: <code>CONNECTION=connection</code> CONNECTION='' indicates access to the local ODBC database server. Absence of the CONNECTION attribute indicates access to the default database server.
DBSPACE	Optional keyword that indicates the storage area for the table. For example: <code>datasource.tablespace</code> <code>DATABASE datasource</code>
KEYS	Indicates how many columns constitute the primary key for the table. Range is 0 to 64. Corresponds to the first <i>n</i> fields in the Master File segment.

Keyword	Description
WRITE	Specifies whether write operations are allowed against the table.
KEYORDER	Optional keyword that specifies the logical sort sequence of data by the primary key; it does not affect the physical storage of data. Used by the adapter for FST. and LST. operations in report requests.

Data Type Support

Data types are specific to the underlying data source.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Syntax

How to Override Default Precision and Scale

```
ENGINE [SQLODBC] SET CONVERSION RESET
ENGINE [SQLODBC] SET CONVERSION format RESET
ENGINE [SQLODBC] SET CONVERSION format [PRECISION pp [ss]]
ENGINE [SQLODBC] SET CONVERSION format [PRECISION MAX]
```

where:

SQLODBC

Indicates the Data Adapter for ODBC. You can omit this value if you previously issued the SET SQLENGINE command.

RESET

Returns any previously specified precision and scale values to the data adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

format

Is any valid format supported by the data source. Possible values are:

INTEGER which indicates that the command applies only to INT columns.

DECIMAL which indicates that the command applies only to DECIMAL columns.

FLOAT which indicates that the command applies only to double precision floating point columns.

pp

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

ss

Is the scale. This is valid with DECIMAL and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. If the scale is not required, you must set mm to 0 (zero).

MAX

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE SQLODBC SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLODBC SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE SQLODBC SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLODBC SET CONVERSION RESET
```

Customizing the ODBC Environment

The Data Adapter for ODBC provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to ODBC.

Syntax How to Issue the TIMEOUT Command

The syntax is

```
ENGINE [SQLODBC] SET TIMEOUT {nn|0}
```

where:

SQLODBC

Indicates the SQLENGINE setting. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a timeout occurs. The default value is 30.

0

Represents an infinite period to wait for response.

CHAPTER 27

Getting Started in Oracle

Topics:

- Preparing the Oracle Environment
- Configuring the Data Adapter for Oracle
- Managing Oracle Metadata
- Customizing the Oracle Environment
- Calling an Oracle Stored Procedure Using SQL Passthru

The Data Adapter for Oracle allows applications to access Oracle data sources. The adapter converts data or application requests into native Oracle statements and returns optimized answer sets to the requesting program.

Preparing the Oracle Environment

The Data Adapter for Oracle minimally requires the installation of the Oracle Client. The Oracle Client allows you to connect to a local or remote Oracle database server.

Procedure How to Prepare the Oracle Environment on Windows NT/2000

On Windows NT/2000, the Oracle environment is set up during the installation of Oracle.

Procedure How to Prepare the Oracle Environment on UNIX

1. Specify the Oracle Database Instance to access using the UNIX environment variable \$ORACLE_SID. For example:

```
ORACLE_SID=ora901
export ORACLE_SID
```

2. Specify the location of the Oracle database you wish to access using the UNIX environment variable \$ORACLE_HOME. For example, to set the home directory for the Oracle software to /usr/oracle/9.0.1, specify:

```
ORACLE_HOME=/usr/oracle/9.0.1
export ORACLE_HOME
```

3. Specify the path to the Oracle shared library using the UNIX environment variable \$LD_LIBRARY_PATH. For example:

```
LD_LIBRARY_PATH=$ORACLE_HOME/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH
```

Procedure **How to Prepare the Oracle Environment on OS/390 and z/OS**

The configuration for Oracle on OS/390 and z/OS requires that EDASTART JCL is used to allocate Oracle specific variables. This can be done via the EDAENV ddname in the JCL:

```
ORACLE_SID=ORAT
ORACLE_HOME=/usr/lpp/ora9
LIBPATH=$ORACLE_HOME/lib
```

Syntax **How to Prepare the Oracle Environment on OpenVMS**

When a site creates an Oracle SID, the Oracle software automatically generates a DCL setup script so users and products such as iWay can properly invoke the environment.

The specification for the location of the Oracle setup file is:

```
$@disk:[oracle_root.DB_oraclesiddb]ORAUSER_oraclesiddb.COM
```

where:

disk

Is the disk on which Oracle is installed.

oracle_root

Is the root directory of the Oracle installation.

oraclesiddb

Is the name of the Oracle database. This name does not need to match the Oracle SID.

EDAENV.COM is an optional OpenVMS iWay file that is invoked at server start up to issue DCL commands such as calls to DBMS setup files. Due to Oracle's use of symbols and job logicals, the proper way to invoke Oracle's setup file is only via the EDAENV.COM file.

By default, EDAENV.COM does not exist and must be manually created in the [.BIN] directory of EDACONF. In this case, and in its simplest form, EDAENV.COM will contain a single line of syntax that specifies the call to the Oracle setup file.

Example **Specifying the \$ORACLE_SID on OpenVMS**

```
$@3$DKB0:[ORACLE.DB_ORA901]ORAUSER_ORA901.COM
```

Connecting to a Remote Oracle Database Server

Using the standard rules for deploying the Oracle Client, the server supports connections to:

- Local Oracle database servers.
- Remote Oracle database servers. To connect to a remote Oracle database server, the Oracle `tnsnames.ora` file on the source machine must contain an entry pointing to the target machine and the listening process must be running on the target machine.

Once you are connected to an Oracle database server, that server may define Oracle DATABASE LINKs that can be used to access Oracle tables on other Oracle database servers.

XA Support

Read/write applications accessing Oracle data sources are able to perform transactions managed in XA-compliant mode.

To activate the XA Transaction Management feature, the server has to be configured in Transaction Coordination Mode, using the Web console configuration functions. Using Transaction Coordination Mode guarantees the integrity of data modification on all of the involved DBMSs and protects part of the data modifications from being committed on one DBMS and terminated on another.

For complete documentation on XA compliance, see the XA Support appendix.

Configuring the Data Adapter for Oracle

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to an Oracle database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Manually add the command in the global server profile (edasprof.prf) or in a user profile (user.prf).
- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the global server profile. For more information, see *Declaring Oracle Connection Attributes from the Web Console*.

You can declare connections to more than one Oracle database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to Oracle Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Syntax How to Declare Connection Attributes Manually

Explicit authentication. The user ID and password are explicitly specified for each connection and passed to Oracle, at connection time, for authentication. The syntax is

```
ENGINE [SQLORA] SET CONNECTION_ATTRIBUTES [connection]
/user_ID,password
```

Password passthru authentication. The user ID and password are explicitly specified for each connection and passed to Oracle, at connection time, for authentication. This option requires that the server be started with security off. The syntax is

```
ENGINE [SQLORA] SET CONNECTION_ATTRIBUTES [connection]
/
```

Trusted authentication. The adapter connects to Oracle as a Windows login using the credentials of the Windows user impersonated by the server data access agent. The syntax is

```
ENGINE [SQLORA] SET CONNECTION_ATTRIBUTES [connection]
/,
```

where:

SQLORA

Indicates the Data Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is a logical name (or service (TNS) name) used to identify this particular set of attributes. If you plan to have only a local connection to Oracle, this parameter is optional. If not specified, the local database server serves as the default connection.

user_ID

Is the primary authorization ID by which you are known to Oracle.

password

Is the password associated with the primary authorization ID.

Example Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command allow the application to access the Oracle database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLORA SET CONNECTION_ATTRIBUTES SAMPLESERVER/MYUSER,PASS
```

The following SET CONNECTION_ATTRIBUTES command connects to the Oracle database server named SAMPLESERVER using Password Passthru authentication:

```
ENGINE SQLORA SET CONNECTION_ATTRIBUTES SAMPLESERVER/
```

The following SET CONNECTION_ATTRIBUTES command connects to a local Oracle database server using operating system authentication:

```
ENGINE SQLORA SET CONNECTION_ATTRIBUTES /,
```

Reference Declaring Oracle Connection Attributes From the Web Console

Attribute	Description
TNS name	<p>Is the service (TNS) name used as a connect descriptor to a Oracle database server across the network. It must point to a valid entry in the tnsnames file.</p> <p>If you plan to have only one connection to Oracle, this parameter is optional. If not specified, the local database server serves as the default connection.</p>
Security	<p>There are three methods by which a user can be authenticated when connecting to an Oracle database server:</p> <p>Explicit. The user ID and password are explicitly specified for each connection and passed to Oracle, at connection time, for authentication.</p> <p>Password Passthru. The user ID and password received from the client application are passed to Oracle, at connection time, for authentication. This option requires that the server be started with security off.</p> <p>Trusted. The adapter connects to Oracle as a Windows login using the credentials of the Windows user impersonated by the server data access agent.</p>
User	Is the primary authorization ID by which you are known to Oracle.
Password	Is the password associated with the primary authorization ID.

Connection Syntax in Earlier Server Releases

The following table lists the syntax for the SET CONNECTION_ATTRIBUTES command according to the different server releases:

Server Release	Explicit Authentication	Password Passthru Authentication	Trusted Authentication
4.2.1	SQL SQLORA SET USER <i>userid/password[@tns_name]</i>	No SET USER command in profile permitted.	SQL SQLORA SET USER/""
4.3.1	SQL SQLORA SET USER <i>userid/password[@tns_name]</i>	SQL SQLORA SET USER <i>[@tns_name]</i>	SQL SQLORA SET USER <i>/[@tns_name]</i>
5.1.0	ENGINE SQLORA SET CONNECTION_ATTRIBUTES <i>[tns_name]/userid,password</i>	ENGINE SQLORA SET CONNECTION_ATTRIBUTES <i>[tns_name]/</i>	ENGINE SQLORA SET CONNECTION_ATTRIBUTES <i>[tns_name]/,</i>

Overriding the Default Connection

Once connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax How to Change the Default Connection

```
ENGINE [SQLORA] SET DEFAULT_CONNECTION [connection]
```

where:

SQLORA

Indicates the Data Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the service name defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following error message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following error message is issued: FOC1671, Command out of sequence.

Example Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the Oracle database server named SAMPLENAME as the default Oracle database server:

```
ENGINE SQLORA SET DEFAULT_CONNECTION SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax How to Control the Connection Scope

```
ENGINE [SQLORA] SET AUTODISCONNECT ON {FIN|COMMAND}
```

where:

SQLORA

Indicates the Data Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

FIN

Disconnects automatically only after the session has been terminated. This value is the default.

COMMAND

Disconnects automatically after each request. Depending on how often the event occurs, the AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server; it is related to the operating system and the data source.

Managing Oracle Metadata

When the server accesses a data source, it needs to know how to interpret the data that it finds. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Oracle data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Oracle table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Editing Metadata page opens.

To create a synonym, you must have configured the adapter. See *Configuring the Data Adapter for Oracle* for more information.

2. Expand the *Add* folder, expand the adapter folder, and then click a connection. The right pane displays table and view selection options:

Select All Tables/Views. Select this radio button to create synonyms for all tables and views. This value is the default.

Filter by Name, Owner and Table Type. Select this radio button to filter the tables or views for which to create synonyms.

Selecting this option adds the following:

The screenshot shows a dialog box with a dark blue background. At the top, there are two radio buttons: the first is labeled "Select all Tables/Views" and is unselected; the second is labeled "Filter by Name, Owner and Table Type" and is selected. Below these are three input fields: "Owner" with an empty text box and a sample "- Sample: abc%"; "Table name" with a text box containing "NF%" and a sample "- Sample: abc%"; and "Table type" with three radio buttons: "TABLE" (selected), "VIEW", and "TABLE/VIEW". At the bottom left, there is a button labeled "Select Tables".

Owner. Enter a string for filtering the owners' IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owners' IDs begin with the letters ABC; %ABC to select tables or views whose owners' IDs end with the letters ABC; %ABC% to select tables or views whose owners' IDs contain the letters ABC at the beginning, middle, or end.

Table Name. Enter a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

Table Type. Select one of the following options: Tables (this is the default), Views, Tables/Views.

3. Click *Select Tables*. All tables that meet the specified criteria are displayed:

Prefix: SET CONVERSION LONGCHAR TEXT (for Web Focus) ▼

Select Application Directory: ▼

All Select

Check	Default Synonym Name	Owner/Schema	Table Name	Type
<input type="checkbox"/>	<input type="text" value="NF29001"/>	EDAQA	NF29001	TABLE
<input type="checkbox"/>	<input type="text" value="NF29002"/>	EDAQA	NF29002	TABLE
<input type="checkbox"/>	<input type="text" value="NF29003"/>	EDAQA	NF29003	TABLE
<input type="checkbox"/>	<input type="text" value="NF29004"/>	EDAQA	NF29004	TABLE
<input type="checkbox"/>	<input type="text" value="NF29005"/>	EDAQA	NF29005	TABLE
<input type="checkbox"/>	<input type="text" value="NF29006"/>	EDAQA	NF29006	TABLE
<input type="checkbox"/>	<input type="text" value="NF29007"/>	EDAQA	NF29007	TABLE
<input type="checkbox"/>	<input type="text" value="NF29008"/>	EDAQA	NF29008	TABLE
<input type="checkbox"/>	<input type="text" value="NF29009"/>	EDAQA	NF29009	TABLE
<input type="checkbox"/>	<input type="text" value="NF29012"/>	EDAQA	NF29012	TABLE
<input type="checkbox"/>	<input type="text" value="NF29013"/>	EDAQA	NF29013	TABLE

4. If you have tables with identical table names, assign a prefix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave field blank.

5. From the SET CONVERSION LONGCHAR drop-down list, select: Text, Alpha, or BLOB. The default value is Text. See *Controlling the Mapping of Large Character Data Types* for more information.
6. From the Select Application Directory drop-down list, select a directory. The default value is baseapp.
7. Complete your table or view selection:

To select all tables or views in the list, click **All**.

To select specific tables or views, click the corresponding check boxes.

8. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.

9. Click *Create Synonym*. Synonyms are created and added under the specified application directory.

Reference **Managing Synonyms**

In the navigation pane, click the name of the synonym to access the following options:

Sample Data	Retrieves up to 20 rows from the associated data source.
Properties	Displays a graphic representation of the synonym and enables you to edit its metadata.
Edit Master File	Enables you to manually edit the synonym's Master File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Master File, to update the synonym.
Edit Access File	Enables you to manually edit the synonym's Access File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Access File, to update the synonym.
Refresh	Regenerates the synonym. Use this option if structural changes were made to the data source.
Drop	Deletes the synonym.
Copy to	Copies the synonym to another application directory. Click the target directory from the resulting list.
Move to	Moves the synonym to another application directory. Click the target directory from the resulting list.

Syntax **How to Create a Synonym Manually**

```
CREATE SYNONYM app/synonym FOR table_view DBMS SQLORA [AT connection]  
[NOCOLS] [AT '']
```

END

where:

app

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, then this application name must not be used.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema) and the database link name as follows:

```
[owner.] table [ @databaseLink ]
```

SQLORA

Indicates the Data Adapter for Oracle.

AT *connection*

Is the service name as previously specified in a SET CONNECTION_ATTRIBUTES command. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

This parameter is optional. If specified, the CONNECTION attribute is added to the Access File.

AT ' '

Adds CONNECTION=' ' in the Access File. This indicates a connection to the local Sybase database server.

NOCOLS

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.NF29004 DBMS SQLORA AT ORA901 NOCOLS
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION ,SUFFIX=SQLORA,$
SEGNAME=SEG1_4 ,SEGTYPE=S0 ,$
FIELD=DIVISION4 ,DIVISION4 ,I9 ,I4 ,MISSING=OFF ,$
FIELD=DIVISION_NA4,DIVISION_NA4 ,A25 ,A25 ,MISSING=ON ,$
FIELD=DIVISION_HE4,DIVISION_HE4 ,I9 ,I4 ,MISSING=ON ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4, TABLENAME=EDAQA.NF29004,
CONNECTION=ORA901, KEYS=1, WRITE=YES,$
```

Reference Access File Keywords

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the Oracle table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: <code>TABLENAME=[owner.]table [@databaselink]</code>
CONNECTION	Indicates a previously declared connection. The syntax is: <code>CONNECTION=connection</code> CONNECTION=' ' indicates access to the local Oracle database server. Absence of the CONNECTION attribute indicates access to the default database server.

Accessing Multiple Database Servers in One SQL Request

To access a remote Oracle table using DATABASE LINKs, the following conditions must exist:

- The Oracle database server to which you are connected must have a valid DATABASE LINK defined.
- The TABLENAME attribute in the Access File for the Oracle table to be queried must have the following format:

```
TABLENAME=[owner.]table@databaselink
```

where:

`owner`

Is the user ID by default. It can consist of a maximum of 30 characters. Oracle prefers that the value be uppercase.

`table`

Is the name of the table or view. It can consist of a maximum of 30 characters.

`databaselink`

Is the valid DATABASE LINK name defined in the currently connected Oracle database server.

This format for TABLENAME can be placed in the Access File manually or using the CREATE SYNONYM command. For example:

```
CREATE SYNONYM filename FOR owner.table @databaselink DBMS SQLORA
```

Once you have met the above conditions, all requests for the table will be processed on the remote Oracle database server specified using the DATABASE LINK name. Using this method is another way to access multiple remote servers in one SQL request.

Data Type Support

The following table lists how the server maps Oracle data types. Note that you can:

- Control the mapping of large character data types.
- Change the mapping of variable-length data types.
- Change the precision and scale of numeric columns.

Be sure to review those options carefully as they affect the mapping of data types.

Oracle Data Type	Data Type		Remarks
	Usage	Actual	
CHAR (<i>n</i>)	<i>An</i>	<i>An</i>	<i>n</i> is an integer between 1 and 2000
NCHAR (<i>n</i>)	<i>An</i>	<i>An</i>	<i>n</i> is an integer between 1 and 2000
VARCHAR (<i>n</i>)	<i>An</i>	<i>An</i>	<i>n</i> is an integer between 1 and 4000
VARCHAR2 (<i>n</i>)	<i>An</i>	<i>An</i>	<i>n</i> is an integer between 1 and 4000
NVARCHAR2 (<i>n</i>)	<i>An</i>	<i>An</i>	<i>n</i> is an integer between 1 and 4000
RAW (<i>n</i>)	<i>Am</i>	<i>Am</i>	<i>n</i> is an integer between 1 and 2000 $m = 2 * n$
LONG	TX50	TX	Supported through iWay API
LONG RAW	BLOB	BLOB	Supported through iWay API
BLOB			Not supported
CLOB	TX50	TX	Server supports up to 32K with ora_oci=y in edaserve.cfg
NCLOB			Not supported
ROWID	A18	A18	
UROWID			Not supported
MLSLABEL			Not supported

Oracle Data Type	Data Type		Remarks
	Usage	Actual	
BFILE			Not supported
NCLOB			Not supported
DATE	HYYMD	HYYMD	
TIMESTAMP (fractional_seconds_precision) WITH LOCAL TIME ZONE			Not supported New in Oracle9i
INTERVAL YEAR (year_precision) TO MONTH			Not supported New in Oracle9i
INTERVAL DAY (day_precision) TO SECOND (fractional_seconds_precision)			Not supported New in Oracle9i
NUMBER (p, s)	P16	Pp,s	p is an integer between 1 and 31s is an integer between 0 and p
	D8	D20.2	p is an integer between 32 and 37s is an integer between 0 and p
	I4	I11	p is 38. This value is the Oracle default.s is an integer between 0 and p
INTEGER			Not an Oracle built-in data type. Stored in Oracle as NUMBER.
DECIMAL			Not an Oracle built-in data type. Stored in Oracle as NUMBER.
FLOAT DOUBLE PRECISION			Not an Oracle built-in data type. Stored in Oracle as NUMBER.

Controlling the Mapping of Large Character Data Types

The SET parameter CONVERSION LONGCHAR controls the mapping of supported Oracle data types listed below. By default, the server maps these data types as alphanumeric (A). The server data type A supports a maximum of 4096 characters for TABLE/MODIFY and 32768 characters for API applications.

The following table lists data type mappings based on the value of LONGCHAR:

Oracle Data Type	Remarks	LONGCHAR ALPHA or BLOB		LONGCHAR TEXT	
		Usage	Actual	Usage	Actual
CHAR (<i>n</i>)	<i>n</i> is an integer between 1 and 2000	<i>An</i>	<i>An</i>	TX50	TX
VARCHAR (<i>n</i>)	<i>n</i> is an integer between 1 and 4000	<i>An</i>	<i>An</i>	TX50	TX
VARCHAR2 (<i>n</i>)	<i>n</i> is an integer between 1 and 4000	<i>An</i>	<i>An</i>	TX50	TX
RAW (<i>n</i>)	<i>n</i> is an integer between 1 and 2000 $m = 2 * n$	<i>Am</i>	<i>Am</i>	TX50	TX

Syntax

How to Control the Mapping of Large Character Data Types

```
ENGINE [SQLORA] SET CONVERSION LONGCHAR {ALPHA|TEXT|BLOB}
```

where:

SQLORA

Indicates the Data Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

ALPHA

Maps the Oracle data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as alphanumeric (A). This value is the default.

TEXT

Maps the Oracle data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as text (TX). Use this value for WebFOCUS applications.

BLOB

For UNIX, Windows, OpenVMS, and OS/400, is identical to ALPHA. That is, it maps the Oracle data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as alphanumeric (A).

For OS/390 and z/OS, maps the Oracle data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as binary large object (BLOB).

Controlling the Mapping of Variable-Length Data Types

The SET parameter VARCHAR controls the mapping of the Oracle data types VARCHAR, VARCHAR2, and NVARCHAR2. By default, the server maps these data types as alphanumeric (A).

The following table lists data type mappings based on the value of VARCHAR:

Microsoft SQL Server Data Type	Remarks	VARCHAR ON		VARCHAR OFF	
		Usage	Actual	Usage	Actual
VARCHAR (<i>n</i>)	<i>n</i> is an integer between 1 and 4000	<i>AnV</i>	<i>AnV</i>	<i>An</i>	<i>An</i>
VARCHAR2 (<i>n</i>)	<i>n</i> is an integer between 1 and 4000	<i>AnV</i>	<i>AnV</i>	<i>An</i>	<i>An</i>
NVARCHAR2 (<i>n</i>)	<i>n</i> is an integer between 1 and 4000	<i>AmV</i>	<i>AmV</i>	<i>Am</i>	<i>Am</i>

Syntax

How to Control the Mapping of Variable-Length Data Types

`ENGINE [SQLORA] SET VARCHAR {ON|OFF}`

where:

SQLORA

Indicates the Data Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Maps the Oracle data types VARCHAR, VARCHAR2, and NVARCHAR2 as variable-length alphanumeric (*AnV*).

OFF

Maps the Oracle data types VARCHAR, VARCHAR2, and NVARCHAR2 as alphanumeric (A). This value is the default.

Considerations for the CHAR and VARCHAR2 Data Types

Special attention must be paid to CHAR and VARCHAR2 data types. When you compare a CHAR data type column to a VARCHAR2 data type column, where the only difference is additional trailing spaces in the CHAR data type column, Oracle treats the column values as different.

The SET parameter ORACHAR lets you specify which of the two data types will be used for inserting, updating, and retrieving data.

If you create the tables outside of the server, we recommend that you use either CHAR or VARCHAR2 data types, but not both. If you create a table with both data types, you might not be able to retrieve the data you inserted due to Oracle's comparison mechanism. When inserting data into VARCHAR2 columns outside of the server, do not insert any trailing spaces.

If you use the server to generate Oracle tables and retrieve data, you will not encounter this problem, since the data type being used will be either CHAR or VARCHAR2, depending upon the ORACHAR setting.

Syntax

How to Set ORACHAR

```
ENGINE [SQLORA] SET ORACHAR {FIX|VAR}
```

where:

SQLORA

Indicates the Data Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

FIX

Uses the CHAR data type.

VAR

Uses the VARCHAR2 data type. This value is the default.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Syntax

How to Override Default Precision and Scale

```
ENGINE [SQLORA] SET CONVERSION RESET  
ENGINE [SQLORA] SET CONVERSION format RESET  
ENGINE [SQLORA] SET CONVERSION format [PRECISION pp [ss]]  
ENGINE [SQLORA] SET CONVERSION format [PRECISION MAX]
```

where:

SQLORA

Indicates the Data Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

RESET

Returns any previously specified precision and scale values to the data adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

format

Is any valid format supported by the data source. Possible values are:

INTEGER which indicates that the command applies only to INT columns.

DECIMAL which indicates that the command applies only to DECIMAL columns.

FLOAT which indicates that the command applies only to double precision floating point columns.

pp

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

ss

Is the scale. This is valid with DECIMAL and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. If the scale is not required, you must set *ss* to 0 (zero).

MAX

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE SQLORA SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLORA SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE SQLORA SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLORA SET CONVERSION RESET
```

Considerations for the NUMBER Data Type

When working with the NUMBER data type, where the precision is between 32 and 37, by default the NUMBER data type is mapped to the server data type double float (D), with a precision of 20 and a scale of 2.

When working with the NUMBER data type, where the precision is 38, by default the NUMBER data type is mapped to the server data type Integer (I), with a display length of 11.

To override the precision of the NUMBER data type, where the precision is between 32 and 38, use the ORANUMBER setting.

Syntax

How to Set ORANUMBER

```
ENGINE [SQLORA] SET ORANUMBER {COMPAT|DECIMAL}
```

where:

[SQLORA](#)

Indicates the Data Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

[COMPAT](#)

Indicates that the NUMBER data type will be mapped to the server data type double float (D), with a precision of 20 and a scale of 2. This is the default.

[DECIMAL](#)

Indicates that the NUMBER data type will be mapped to the server data type decimal (P), with a precision of 33.

Customizing the Oracle Environment

The Data Adapter for Oracle provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Designating a Default Tablespace

You can use the SET DBSPACE command to designate a default tablespace for tables you create. For the duration of the session, the adapter places these tables in the Oracle tablespace that you identify with the SET DBSPACE command. If the SET DBSPACE command is not used, Oracle uses the default tablespace for the connected user.

Syntax How to Set DBSPACE

```
ENGINE [SQLORA] SET DBSPACE tablespace
```

where:

SQLORA

Indicates the Data Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

tablespace

Is a valid tablespace in the database.

Note: This command will affect only CREATE FILE requests issued by Table Services. It does not affect Passthru CREATE TABLE commands.

Syntax How to Designate a Default Storage Space for Tables

```
ENGINE [SQLORA] SET DBSPACE {datasource.tablespace|DATABASE datasource}
```

where:

SQLORA

Indicates the Data Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

datasource

Is the data source name. The default value is DSNDDB04, a public data source.

tablespace

Is a valid table space name in the data source.

Note: This command will only affect CREATE FILE requests issued by Table Services. It does not affect Passthru CREATE TABLE commands.

Specifying the Block Size for Array Retrieval

The Data Adapter for Oracle supports array retrieval from result sets produced by executing SELECT queries or stored procedures. This technique substantially reduces network traffic and CPU utilization.

Using high values increases the efficiency of requests involving many rows, at the cost of higher virtual storage requirements. A value higher than 100 is not recommended because the increased efficiency it would provide is generally negligible.

Syntax How to Specify the Block Size for Array Retrieval

The block size for a SELECT request applies to TABLE FILE requests, MODIFY requests, MATCH requests, and DIRECT SQL SELECT statements.

```
ENGINE [SQLORA] SET FETCHSIZE n
```

where:

SQLORA

Indicates the Data Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is the number of rows to be retrieved at once using array retrieval techniques. Accepted values are 1 to 5000. The default is 20. If the result set contains a column that has to be processed as a CLOB or a BLOB, the fetchsize value used for that result set is 1.

Syntax How to Specify the Block Size for Inserting Rows

In combination with LOADONLY, the block size for an INSERT applies to MODIFY INCLUDE requests. INSERTSIZE is also supported for parameterized DIRECT SQL INSERT statements.

```
ENGINE [SQLORA] SET INSERTSIZE n
```

where:

SQLORA

Indicates the Data Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is the number of rows to be inserted using array insert techniques. Accepted values are 1 to 5000. The default is 1. If the result set contains a column that has to be processed as a BLOB, the insertsize value used for that result set is 1.

Overriding Default Parameters for Index Space

You can use the SET IXSPACE command to override the default parameters for the Oracle index space implicitly created by the CREATE FILE and HOLD FORMAT SQLORA commands.

Syntax How to Set IXSPACE

```
ENGINE [SQLORA] SET IXSPACE [index-spec]
```

where:

SQLORA

Indicates the Data Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

index-spec

Is the portion of the Oracle CREATE INDEX statement that defines the parameters for the index. It can consist of up to 94 bytes of valid Oracle index space parameters. To reset the index space parameters to their default values, issue the SET IXSPACE command with no parameters.

Note: Refer to the Oracle documentation for more information on this command.

The long form of SQL Passthru syntax for commands exceeding one line is:

```
ENGINE SQLORA
SET IXSPACE index-spec
END
```

For example, to specify the NOSORT, NOLOGGING, and TABLESPACE portions of the Oracle CREATE INDEX statement, enter the following commands:

```
ENGINE SQLORA
SET IXSPACE NOSORT NOLOGGING
TABLESPACE TEMP
END
```

Note: This command will only affect CREATE INDEX requests issued by CREATE FILE and HOLD FORMAT SQLORA commands. It does not affect Passthru CREATE INDEX commands, for example:

```
ENGINE SQLORA SET IXSPACE TABLESPACE tablespace_name
TABLE FILE table_name
PRINT *
ON TABLE HOLD AS file_name FORMAT SQLORA
END
```

Activating NONBLOCK Mode

The Data Adapter for Oracle has the ability to issue calls in NONBLOCK mode. The default behavior is BLOCK mode.

This feature allows the adapter to react to a client request to cancel a query while the adapter is waiting on engine processing. This wait state usually occurs during SQL parsing, before the first row of an answer set is ready for delivery to the adapter or while waiting for access to an object that has been locked by another application.

Syntax

How to Activate NONBLOCK Mode

```
ENGINE [SQLORA] SET NONBLOCK {0|n}
```

where:

SQLORA

Indicates the Data Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is a positive numeric number. A value of 0, the default, means that the adapter will operate in BLOCK mode. A value of 1 or greater activates the NONBLOCK calling and specifies the time, in seconds, that the adapter will wait between each time it checks to see if the:

- Query has been executed.
- Client application has requested the cancellation of a query.
- Kill Session button on the Web Console is pressed.

Note: A value of 1 or 2 should be sufficient for normal operations.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Syntax How to Obtain the Number of Rows Updated or Deleted

```
ENGINE [SQLORA] SET PASSRECS {ON|OFF}
```

where:

SQLORA

Indicates the Data Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. This value is the default.

OFF

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Specifying the Maximum Number of Parameters for Stored Procedures

You can use the SET SPMAXPRM command to specify the maximum number of input parameters that can be associated with any Oracle stored procedure.

Syntax How to Set SPMAXPRM

```
ENGINE [SQLORA] SET SPMAXPRM nnn
```

where:

SQLORA

Indicates the Data Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

nnn

Is the maximum number of parameters that can be passed to any stored procedure available to be run in this client session. The default is 256.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to Oracle.

Syntax How to Issue the TIMEOUT Command

The syntax is

```
ENGINE [SQLORA] SET TIMEOUT {nn|0}
```

where:

SQLORA

Indicates the SQLENGINE setting. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a timeout occurs. The default value is 30.

0

Represents an infinite period to wait for response.

Calling an Oracle Stored Procedure Using SQL Passthru

Using SQL Passthru is supported for Oracle stored procedures. These procedures need to be developed within Oracle using the CREATE PROCEDURE command.

Example Calling a Stored Procedure

The supported syntax to call a stored procedure is shown below. It is recommended that you use the syntax below instead of the previously supported CALLINF, CALLORA, CALLSYB, and CALLDB2 syntax.

```
ENGINE SQLORA  
EX SAMPLE PARM1,PARM2,PARM3...;  
TABLE ON TABLE PCHOLD  
END
```

The server supports invocation of stored procedures written according to the following rules:

- Only scalar input parameters (IN or IN OUT) are allowed, but not required.
- Output parameters are not allowed.
- A cursor must be defined with:
 - The TYPE statement in a PACKAGE or PROCEDURE.
 - An associated record layout of the answer set to be returned.

- The cursor must be opened in the procedure.
- No fetching is allowed in the stored procedure. The Data Adapter for Oracle fetches the answer set.
- Any error messages must be issued using the RAISE APPLICATION ERROR method.

Note: Any application error that is issued by the stored procedure is available in the server variable &ORAMSGTXT.

Example Oracle Stored Procedure

```
CREATE OR REPLACE PACKAGE pack1 AS
TYPE nfrectype IS RECORD (
employee NF29005.EMPLOYEE_ID5%TYPE,
ssn5     NF29005.SSN5%TYPE,
l_name   NF29005.LAST_NAME5%TYPE,
f_name   NF29005.FIRST_NAME5%TYPE,
birthday NF29005.BIRTHDATE5%TYPE,
salary   NF29005.SALARY5%TYPE,
joblevel NF29005.JOB_LEVEL5%TYPE);
TYPE nfcurtype IS REF CURSOR RETURN nfrectype ;
PROCEDURE procl(c_saltable IN OUT nfcurtype);
END pack1 ;
/
CREATE OR REPLACE PACKAGE BODY pack1 AS
PROCEDURE procl (c_saltable IN OUT nfcurtype)
IS
BEGIN
OPEN c_saltable FOR SELECT
EMPLOYEE_ID5,SSN5,LAST_NAME5,FIRST_NAME5,BIRTHDAT
E5,SALARY5,JOB_LEVEL5 FROM NF29005;
END procl ; -- end of procedure
END pack1; -- end of package body
/
```

Calling an Oracle Stored Procedure Using SQL Passthru

CHAPTER 28

Getting Started in PeopleSoft

Topics:

- Preparing to Configure the Data Adapter for PeopleSoft
- Preparing the Server Environment for Adapter Configuration
- Configuring the Data Adapter for PeopleSoft
- Using the PeopleSoft Metadata and Security Administrator
- Managing PeopleSoft Metadata
- Security Access Administration
- Administration Reports
- Advanced Topics

The Data Adapter for PeopleSoft provides data access for reporting from the following PeopleSoft record types: SQL tables, SQL views, and query views. The adapter fully enforces PeopleSoft Query tree and low-level security.

The adapter supports both the client-server architecture of PeopleSoft 7.5 and earlier, as well as the newer PeopleSoft Pure Internet Architecture.

The adapter must be configured on the server that will be used for PeopleSoft reporting *prior* to creating any connections to PeopleSoft data sources.

The adapter provides the following three classes of services:

- Authentication services.
- PeopleSoft data source connection management services.
- Metadata administration and data access security services.

Native data source SQL scripts are provided to prepare the following PeopleSoft data constructs for simplified reporting:

- Effective dated records.
- PeopleSoft trees.

For detailed information about the purpose and implementation of these scripts, contact your iWay Software representative.

Preparing to Configure the Data Adapter for PeopleSoft

The following software is required on the server to support the Data Adapter for PeopleSoft:

- Any PeopleSoft PeopleTools 7.x or 8.x based application or PeopleTools Enterprise.
All PeopleSoft applications are built on the PeopleTools metadata platform, which is the framework for adapter integration.
- An RDBMS license for Oracle, SQL Server, or DB2 Universal data source as applicable.
- An appropriate RDBMS connectivity product on the computer where you will install the Data Adapter for PeopleSoft server agent, for example Oracle SQL*Net for Oracle data source connectivity. This is to provide access to a PeopleSoft data source server for operational data access.

Note: For information about using the Data Adapter for PeopleSoft with versions of software other than those listed in this topic, contact your iWay Software representative.

Security

The Data Adapter for PeopleSoft offers the option of enforcing PeopleSoft security. You can keep security off during the initial phases of testing and then turn security on before going to production. Alternatively, you can turn security off to verify that report results are correct or for other business purposes. Remember that PeopleSoft row-level security can significantly slow down the run time of certain reports.

The Data Adapter for PeopleSoft initially authenticates with PeopleSoft using the PeopleSoft user ID and password entered on the login screen. If successful, the adapter attaches to the RDBMS using an access ID of your choice. Typically, this access ID is either the default PeopleSoft access ID (often it is SYSADM), or it is an RDBMS ID dedicated to the adapter. In either case, the access ID must have read access to the following PeopleSoft tables and views:

PSDBFIELD	PSRECDEFN	PS_SCRTY_ACC_GRP
PSOPTIONS	PSRECFIELD	PS_TREE_ACCESS_VW
PSOPRCLS	PSTREEDEFN	XLATTABLE
PSOPRDEFN	PSTREENODE	PSDBFLDLABL (release 8.x)
PS_FACT_CTRL_TBL	PS_FACT_MAP_TBL	PS_HIER_CTRL_TBL
PSPRCSRQST	PSPRCSQUE (release 8.x)	PSCLASSDEFN (release 8.x)

In addition, the access ID must have read access to any PeopleSoft records that you wish to report against. If your PeopleSoft records have row-level security, this ID must also have read access to the security search records associated with each.

Note: PS_FACT_CTRL_TBL, PS_FACT_MAP_TBL, and PS_HIER_CTRL_TBL are required only if installing OLAP-enabled data marts. The PSPRCSRQST and PSPRCSQUE tables are required only for Process Scheduler integration. If you are using this integration, these two files require write privileges.

Location of the Data

The Data Adapter for PeopleSoft may be co-located with the PeopleSoft back-end data source. Alternatively, it may be on any platform that has appropriate RDBMS connectivity to the PeopleSoft data. For efficiency, it is recommended to use a co-location data architecture whenever this is feasible.

Usually, you choose to co-locate *unless*:

- The platform with the RDBMS has limited resources (for example, hard disk capacity, memory, or processor power). In this case it is necessary to limit additional resource drain.
- The design of your system prohibits installing software on the RDBMS Server.

The installation details change to support the data architecture you choose. For more information, see *Configuring the Data Adapter for PeopleSoft* on page 28-15.

Configuration Worksheet

Gather the configuration information required to configure the server before starting the process. Use the following worksheet to record the configuration information and keep it for future reference.

Field Name	Response	Description
DBA		8-character identifier for linking metadata to PeopleSoft security rules. Must be consistent across multiple connections.
PeopleTools Release		Major release of PeopleTools (7.x, 8.1x, or 8.4x)
Security Enabled?		Specifies whether PeopleSoft data security (row-level and access group) is enforced.
Specify Security Access By		Define how to specify security access; by User (OPRID) or permission list (OPRCLASS).
Database Type		Designation for RDBMS. The options are Oracle, MS SQL Server, or DB2 Universal Database.
Database Server		Name of the Database Server (for SQL Server only, not Oracle or DB2).
Database Identifier		Database System Identifier (SID) or Source Name used by the appropriate DBMS client software on the server.
Database Owner		Data source owner ID for the PeopleSoft data source.
Access ID		ID used by PeopleSoft to connect to the RDBMS.
Access Password		Password associated with the access ID.
Authentication		Select <i>Password</i> to always verify the user ID/password. Select <i>Trusted</i> to accept only user ID. Trusted authentication is useful when deploying in previously authenticated environments, such as a Portal.
Enforce Mixed Case IDs		Provides backwards compatibility for sites that previously used case-insensitive version of PeopleSoft. (Yes or No)

Field Name	Response	Description
PeopleSoft Application Server		PeopleSoft application server name or IP address.
Jolt Listener Port		Tuxedo Jolt listener port (default: 9000)
Default Synonym		The type of synonym to default to when creating new metadata. Record Name Record Name with Prefix Record Name with Suffix Table Name Table Name with Prefix Table Name with Suffix
Prefix/Suffix Default Value		Provide a default prefix/suffix value if the default synonym option requires one.

Preparing the Server Environment for Adapter Configuration

The server needs access to the PeopleSoft RDBMS through a native data adapter. The server uses environment settings to locate the proper client connectivity software and a default data source to connect to. These environment settings must be in effect prior to starting your server.

For UNIX operating systems, these environment variables are either in effect when the server is started or they are supplied in the edastart script. For Windows operating systems, these environment variables are set in the registry or through the System Properties tool.

Prior to accessing any PeopleSoft data, you must configure the adapter. If a server is not installed on the appropriate platform, install a server before configuring the adapter.

Configuring the PeopleSoft Application Adapter consists of the following steps:

1. Configuring a server.
2. Starting the server.
3. Adding the RDBMS adapter.

Configuring the Server

For UNIX operating systems, these environment variables are either in effect when the server is started or they are supplied in the edastart script. For Windows operating systems, these environment variables are set in the registry or through the System Properties tool.

Oracle provides a unique feature called a BEQ, or BEQueath, connection that provides improved performance when the data source resides on the same system as the server. To leverage this feature:

1. Be sure the ORACLE_SID is set for the PeopleSoft data source when starting the server.
2. When supplying parameters for your PeopleSoft Connection, leave the data source Identifier field empty.

In addition to properly defining the environment for access to the PeopleSoft data source, configuration steps must take place to enable authentication against a PeopleSoft Application Server. These steps are required only for PeopleSoft Connections that use Password authentication mode against a PeopleSoft 8 application. For those connections that are either TRUSTED authentication or PeopleSoft 7, these additional configuration steps can be skipped.

For a complete description of managing connections and Password versus TRUSTED authentication, please refer to *Configuring the Data Adapter for PeopleSoft* on page 28-15.

The following sections detail the server configuration steps:

1. Modifying Environment Settings for PeopleSoft RDBMS Connectivity
2. Configuring for PeopleSoft 8 Password Authentication

Modifying Environment Settings for PeopleSoft RDBMS Connectivity

Before configuring data connectivity the unique data source identifier must be defined within that data source client software. In Oracle for example, the unique data source identifier is often referred to as a Net Name Service, which is defined through the Net8 Configuration tools. By contrast, for Microsoft SQL Server, it is referred to as a Data Source Name and can be defined within the Data Source Administrator.

Once those data source identifiers have been defined access may be established. Before starting the server, the client software must be available and the required environment variables must be set.

Windows Environments

In Windows environments, the installation, or home directory will be defined in one of two ways:

1. In the system registry. The client software, for example, Oracle Net8, automatically updates the registry during installation.
2. In the system properties. The client software sometimes updates this, but often the administrator will need to open the properties window and add any necessary environment variables manually.

Through the methods described above, Oracle can also have an ORACLE_SID defined, which will enable BEQueath connections. If not using BEQueath connections, and for all other data sources, the above steps will suffice. The data source identifier will be explicitly defined during the configuration of PeopleSoft Connections.

UNIX Environments

In UNIX environments, the installation directory and additional client parameters will be defined in one of two ways:

1. As a variable that has been set and exported to the environment.
2. Explicitly set and exported within the edastart script.

Syntax

How to Modify the Server Start-up Script in UNIX

You must add RDBMS instance and path information in the server start-up script, edastart, that resides in the /bin directory under EDACONF. You can perform this in one of the following ways:

- If the server is on the same physical server (co-located) and the RDBMS is Oracle, you can supply the data source Identifier in edastart. This is the recommended approach, as it improves performance in some environments.

or

- You can use the PeopleSoft Configurator to supply the information at run time.

For all other configurations, you *must* supply the data source identifier to the PeopleSoft Configurator.

Syntax **How to Modify edastart for an Oracle Adapter**

If you are using a server with an Oracle data adapter, use a text editor to add the following syntax to the edastart file immediately following the "export EDACONF" line

```
export ORACLE_SID=your_Oracle_SID
export ORACLE_HOME=your_Oracle_Home
```

where:

your_Oracle_SID

Is the Oracle Server ID for the PeopleSoft data source you are accessing. If you are running your server on the same system as your PeopleSoft data source, this is the required syntax for a local configuration. A local Oracle configuration is also known as a BEQueath connection. For more information about the benefits of this type of connection, see the Oracle or PeopleSoft documentation. If the data source resides on a different system, remove this line from the edastart script.

your_Oracle_Home

Is the Oracle home directory for the PeopleSoft data source you are accessing. Whether your data source is local or not, this is a required configuration setting. If this is not set, the environmental setting is used, if available, when the server is started.

Syntax **How to Modify edastart for a DB2 Universal Database CLI Adapter**

If you are using a server with the DB2 Universal Database CLI adapter, use a text editor to add the following syntax to the edastart file immediately following the "export EDACONF" line

```
export DB2INSTANCE=your_UDB_Instance
export INSTHOME=your_UDB_Instance_Home_Path
export DB2CLIINIPATH=your_CLI_INI_Path
export PATH=$EDACONF/snapinst:$PATH
```

where:

your_UDB_Instance

Is the instance name of the UDB data.

your_UDB_Instance_Home_Path

Is the location of the home directory where you can locate UDB connectivity files.

your_CLI_INI_Path

Is the location of the db2cli.ini file that contains your client connectivity settings.

For both edastart topics above, the PATH variable gets pre-pended with the /snapinst directory within the reporting server configuration path. This addition is required for the interface to locate and execute program scripts.

It is recommended that external security be set to Web Console protected mode (export EDAEXTSEC=wcprotect).

Example UNIX Configuration With Oracle Database

This sample EDASTART for UNIX start-up script resides in the /bin directory under EDACONF. Note the two lines that are required to support a server co-located with the back-end Oracle RDBMS:

```
#!/bin/ksh
#
parms=$*
#
export EDAEXTSEC=wcprotect
#
export ORACLE_SID=H840DMO
export ORACLE_HOME=/rdbms/ora817/app/oracle/product/8.1.7
#
exec /u1/ibi/srv52/home/bin/tscom300.out $parms
```

Configuring for PeopleSoft 8 Password Authentication

To leverage integrated authentication against PeopleSoft 8, a JSCOM3 listener must be configured using the Web Console. When completed, all user requests will pass calls to the PeopleSoft 8 Component Interface architecture for authentication. If the configuration is either not using PeopleSoft 8 or the connections are to be configured with TRUSTED authentication, then this step is not necessary.

Note: At this time, the PeopleSoft Component Interface does not support LDAP, even though the PeopleSoft Internet Architecture does read from LDAP repositories in general. This limitation means that if your PeopleSoft site uses LDAP integration for user authentication only, TRUSTED authentication is possible. To leverage your LDAP repository for WebFOCUS, use any of the standard integration methods available. For details see the WebFOCUS Security and Administration manual.

Before configuring JSCOM3, the following conditions must be met:

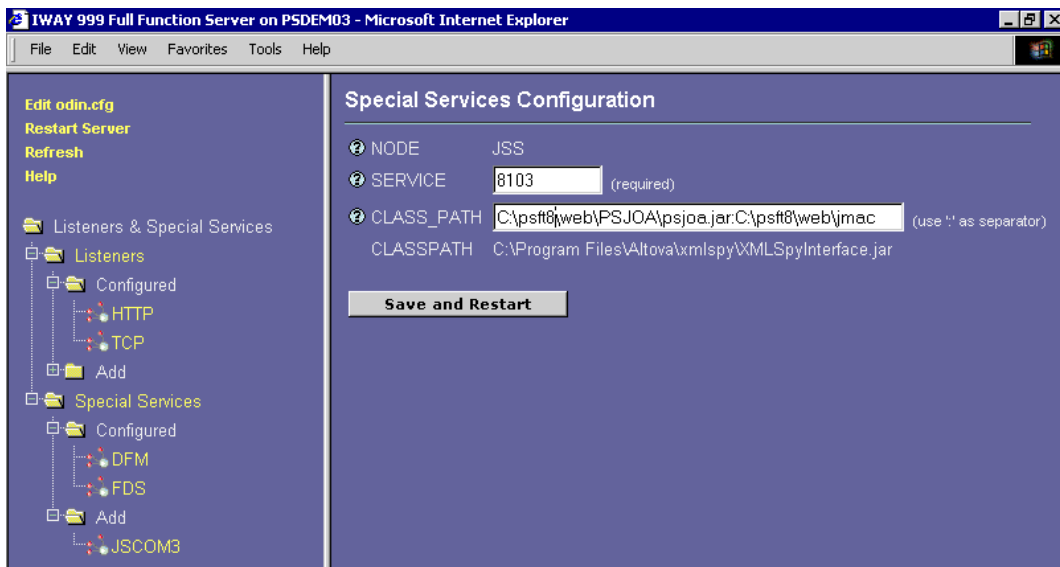
- Ensure that the Publish and Subscribe services are running on the PeopleSoft Application Server. By turning these on, the jolt listener becomes active. Usually, these services are used for application messaging integration requirements, but WebFOCUS for PeopleSoft uses the same technology for authentication. No particular component interface must be accessible for the user to authenticate.
- Several API files delivered with PeopleSoft must be made available to the iWay Reporting Server directory. These files are located in the PeopleSoft file server libraries in the following locations:
 - [PeopleSoft file server]\web\PSJOA\psjoa.jar
 - [PeopleSoft file server]\web\jmac\pstools.properties

Note: With PeopleSoft 8.4 and higher, the pstools.properties file is no longer required.

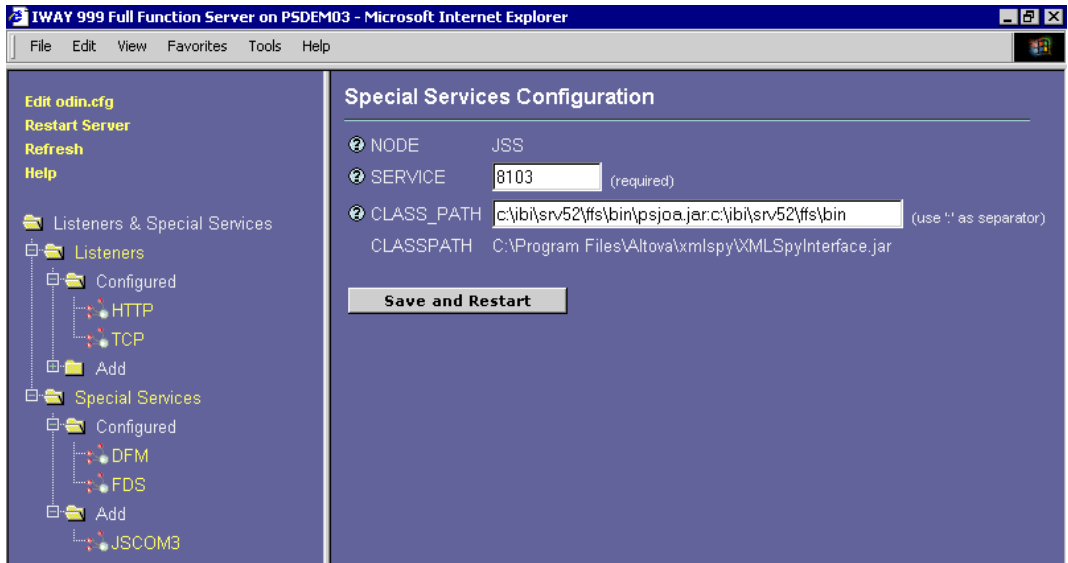
If the iWay Reporting Server is installed on the same computer as your PS Server, then you may configure JSCOM3 to point directly to the files. Otherwise, the files must be copied to the computer with the iWay Reporting Server. In this situation it is recommended (but not required) that they be placed in your EDACONF\bin directory.

To configure the JSCOM3 listener:

1. Open up the Web Console.
2. Select *Listeners* from the menu on the left.
3. Under Special Services, click the + button next to *Add* and select JSCOM3.
4. Enter the JSCOM3 listening port in Services. The examples all use 8103.
5. In the CLASSPATH, enter the psjoa.jar file with the full path and the folder name where pstools.properties resides. If your iWay Reporting Server resides on the same computer as your PeopleSoft file server, your screen should look something like this:



If you copied the files into the EDACONF directory, your screen should look similar to this:



If you do not correctly configure the pstools.properties path or if the file is not in place, the authentication will work, but you will see this message in your jscom3 log file:

```
java.lang.NullPointerException: PSProperties not loaded from file
```

Note: When upgrading the PeopleSoft Application Server after having performed these configuration steps, you must update the PeopleSoft files (psjoa.jar and pstools.properties).

These files are dependant on the minor release level.

Starting the Server

The server can be started in the following ways:

- In Windows, it can be started as a service or with an interactive console.
- In Windows and UNIX, it can be started with or without turning on operating system security integration. Furthermore, the Web Console may be protected to allow access only to designated administrators.

The PeopleSoft solution should usually be started with external security turned off and the Web Console protected. The server will not integrate with operating system based security for authentication and access control. It is uncommon that all PeopleSoft users will have operating system logins since the PeopleSoft Application is already managing them.

The procedures in the following topics describe examples for starting the server with security off and the Web Console protected.

Before starting your server, it must be configured. For details see *Configuring the Server* on page 28-6.

Procedure How to Start the Server on Windows With Security Off

1. Create a new user ID that excludes the following policy privileges:
 - Increase quota.
 - Replace process level token.
 - Act as part of the operating system.
2. Set up the server so that it is started as a service whenever Windows is started. Ensure that it is started under the new user ID, *not* the administrator ID.

Procedure How to Start the Server on UNIX With Security Off

1. Edit the server startup script edastart, located in the server configuration directory, under the bin subdirectory, by adding the environment variable EDAEXTSEC.

Do this by inserting the following command anywhere before the exec line:

```
export EDAEXTSEC = WCPROTECT
```

2. Start the server.

Modifying the Server Profile

The server profile needs to be modified to enable PeopleSoft security integration, both authentication and data access, and to define the proper catalog path for PeopleSoft adapter metadata and control tables.

The server profile, edasprof.prf, resides in the /etc directory under EDACONF. You may either edit this file directly on the server with a text editor of your choice or you may edit the profile through the Web Console. In either case, the server does not need to be recycled for changes to take effect.

Modify the edasprof.prf to do the following:

1. Adding the PeopleSoft adapter paths
2. Enabling integrated PeopleSoft security
3. Additional global settings for PeopleSoft

Adding the PeopleSoft Adapter Paths

The PeopleSoft adapter paths must be added to the cataloged path. There are two directories that are dynamically created under the APPROOT when adding the first connection to a PeopleSoft data source. The directories are snapinst and snapcat. The later has a subdirectory named bak added but this is not catalogued.

We recommend using the APP PATH logic for adding these new directories to the server profile as opposed to using EDAPATH. However, both methods are valid and supported.

Syntax **How to Add the PeopleSoft Adapter Directories as Application Namespaces**

```
APP PATH snapinst snapcat
```

Syntax **How to Add the PeopleSoft Adapter Directories as EDAPATH****For UNIX**

```
APP DISABLE
SET EDAPATH = APPROOT/snapinst: -
              APPROOT/snapcat:
```

where:

```
APPROOT
```

Is the application directory. The default location is home/iadmin/ibi/apps.

For Windows

```
APP DISABLE
SET EDAPATH = APPROOT\snapinst; -
              APPROOT\snapcat;
```

where:

```
APPROOT
```

Is the application directory. The default path is ibi\apps.

Enabling Integrated PeopleSoft Security

In order to gain client access, the PeopleSoft security procedure needs to be called. For easy administration, we recommend calling that procedure globally through the server profile, but it may alternatively be called in a user profile or a within a procedure.

There are three main types of security that are supported through the adapter: trusted access, integrated PeopleSoft authentication, and integrated data access. During the connection configuration steps the administrator decides how the integrated PeopleSoft authentication and data access will be defined.

At a minimum, the PeopleSoft security procedure must be called with USERID as a parameter. This is also the requirement for a connection configured with Trusted authentication. For connections configured with Password authentication, both a user ID and password must be supplied to the procedure call.

Syntax **How to Enable PeopleSoft Security With Trusted Authentication**

```
-SET &UID = CNCTUSR('A30');
EX PSLOGIN USERID=&UID
```

Syntax **How to Enable PeopleSoft Security With Password Authentication**

EX PSLOGIN

Note: The above configuration options for the PeopleSoft security procedure are recommendations. This procedure could also be called with hard-coded USERID and PASSWD values. This may be useful if all users are to receive the same privileges or you have a need to use User Profiles on the server.

Additional Global Settings for PeopleSoft

To ensure the proper functionality of the PeopleSoft adapter's metadata generation process and remove the potential for incorrect report syntax parsing the following two additional settings should be globally defined in the server profile.

Syntax **How to Define Additional Global Settings for PeopleSoft**

SET SYNONYM=BASIC
SET FIELDNAME=NOTRUNC

Note: The FIELDNAME setting is not required, but is a highly recommended setting. By default the FOCUS language makes assumptions when interpreting procedures. One such assumption allows a user to type a partial field name instead of a complete field name (i.e. FOCUS assumes EMPL to be an abbreviation for EMPLID). As a best practice, this setting will prevent incorrect assumptions during interpretation.

Example **Modifying the Server Profile (edasprof.prf)**

The following is a sample server profile. The characters -* in the first two print positions denote a comment. The corresponding comment lines in your file might be slightly different.

```

_*****
-* Profile generated on 1 January 2003 at 13:15:06
_*****
-*
APP ENABLE
APP PATH snapinst snapcat ibisamp
-*
-*
SET FIELDNAME=NOTRUNC
SET SYNONYM=BASIC
-*
SET &UID=CNCTUSR ('A30');
EX PSLOGIN USERID=&UID

```

Note: You must remove any data source statements that were added to edasprof.prf during installation of the server.

For example:

```
ENGINE SQLORA SET CONNECTION_ATTRIBUTES H84DEV/sysadm,FE23E82KSF
```

Adding the RDBMS Adapter

The PeopleSoft Application Adapter is an enhanced version of the underlying data source adapter. The data source adapter must be added into the server configuration.

When an RDBMS data adapter is added, configuration parameters do not need to be provided. This is because those connection parameters are stored in the server profile which is accessible to any user or application connecting to the server at all times.

For most applications, the data access is then secured through procedure logic and operating system security. With PeopleSoft, this is not possible as there may be very large numbers of users who do not have operating system login IDs. Instead, these configuration parameters are supplied later when configuring connections to PeopleSoft data sources.

As a recommended practice, when adding the new data adapter, you may initially supply the configuration parameters, test data source connectivity, and then delete the connection. This process may save significant time later on, when attempting to create a connection to the PeopleSoft data source, but it not required.

Example Adding an Oracle 8.1 Data Adapter

The Web Console provides graphical point and click tools to add and test the data adapter.

1. Open the Web Console.
2. Click *Data Adapters*. A new browser window opens.
3. From the Adapters tree, select *Add*, then *Oracle*, and select *version 8.1.x*.
4. Without supplying Oracle configuration parameters, click *Configure*. The Oracle 8.1 adapter is added to your server configuration.

Configuring the Data Adapter for PeopleSoft

The Web Console is used as the central management tool for adding, removing, and editing existing connections to PeopleSoft data sources. A connection is defined as the named set of configuration parameters required to connect, or gain data access to, a single PeopleSoft data source. The connection parameters define where to locate the data source, the version of the PeopleSoft system, how the metadata is managed, and which PeopleSoft users have reporting access.

Managing Multiple Connections to PeopleSoft Data Sources

Multiple connections to PeopleSoft data sources are possible from within one server configuration. However, depending on your reporting requirements, it may be optimal to create separate configurations for each data source.

Creating multiple connections using the same server configuration is a good option under the following circumstances:

- The site has two or more PeopleSoft application suites and must perform significant combined reporting. A typical example is a site that has both PeopleSoft HRMS and PeopleSoft Financials applications where you must display data from both systems in a single report.
- Two or more PeopleSoft data sources that must be accessed reside on the same system. If the data sources must be accessed for display in a single report output and if the server is located on the same computer system, this architecture can provide optimal performance.
- Two or more PeopleSoft data sources reside in different locations with requirements for moderate to significant consolidated reporting. This architecture provides optimal effectiveness in report development, system maintenance, and performance.

Create separate configurations instead of using multiple connections in these situations:

- Only a single PeopleSoft production data source must be supported. Each development and test instance should have a separate configuration.
- Multiple PeopleSoft data sources exist with only a limited requirement for combining data in reports.

The remaining instructions in this topic assumes a single configuration directory structure for one or more data source connections.

Adding the First PeopleSoft Connection

Although multiple PeopleSoft connections can be created, the first connection requires slightly different management steps. The reason for this is that before any connections to PeopleSoft data sources can occur, a base layer of application metadata is created and configured. This occurs transparently for the Administrator, but is a slightly different connection management process than when adding additional connections.

To create the first connection:

- Select and enter a DBA user.

The DBA user is the 8-character alphanumeric value that is used as a metadata access key. Subsequent access to the Web Console automatically sets this value. Users outside of the Web Console do not have metadata access without first logging in to PeopleSoft through this integration.

- Add a PeopleSoft Connection. For details, see *How to Add the First PeopleSoft Connection* on page 28-17.

Procedure How to Add the First PeopleSoft Connection

1. Open the Web Console.

You can access the Web Console through a Web browser by opening the URL to the server's HTTP listener port. The default location on a local server is `http://localhost:8121/`.

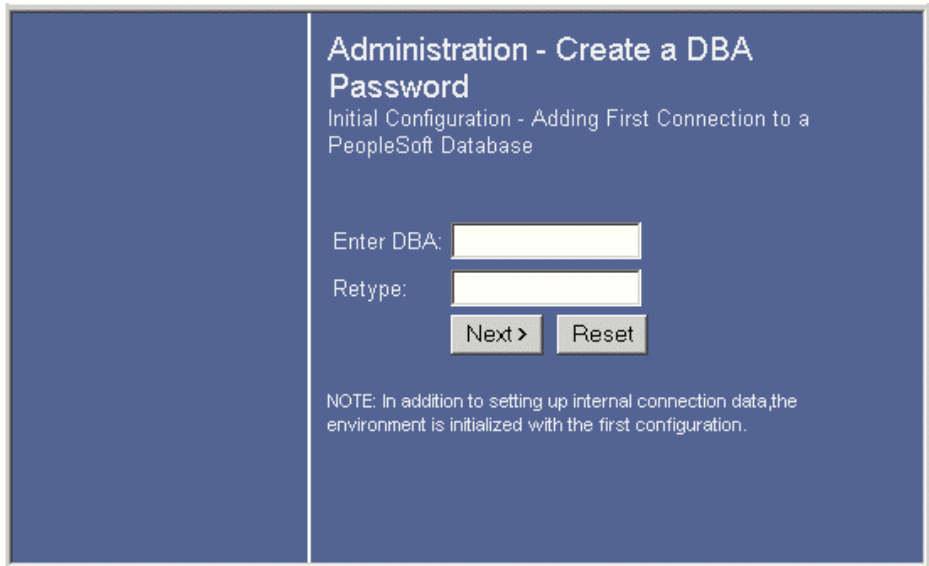
Note: If the Web Console is set to Protected mode, you must supply a Web Console administrator ID and password to gain access.

2. Select *Data Adapters* in the main window.

The Configuring Data Adapters window opens.

3. Select *PeopleSoft* in the Add folder and click *Configure*.

The Administration - Create a DBA Password window opens:



Administration - Create a DBA Password

Initial Configuration - Adding First Connection to a PeopleSoft Database

Enter DBA:

Retype:

NOTE: In addition to setting up internal connection data, the environment is initialized with the first configuration.

4. Enter an up to eight-character DBA in the Enter DBA field and the Retype field, then click *Next>*.

The Administration - Create a New PeopleSoft Connection window opens.

5. Enter the necessary information and click *Next>*. Most of the parameters can be changed in the future, but the connection cannot be created without initially supplying correct data source connectivity information. For details on the required information, see *Administration - Create a New PeopleSoft Connection Window* on page 28-19.

Processing may take several minutes to complete. Once completed, you may begin managing your new PeopleSoft Connection by adding synonyms and administering security access.

Reference Administration - Create a New PeopleSoft Connection Window

Administration - Create a New PeopleSoft Connection

Description

PeopleTools Release 7.x 8.1x 8.4x

Security Parameters

Security Enabled? Yes No

Specify Security Access By

Connection Parameters

Database Type

Database Identifier

Access ID

Access Password

Database Owner

Authentication Options

Authentication

Enforce Mixed Case IDs Yes No

Application Server

Jolt Listener Port

Synonym Options

The Administration - Create a New PeopleSoft Connection window contains the following fields/options:

Description

Is a description that identifies the current server configuration.

PeopleTools Release

Is the major PeopleSoft PeopleTools release number to access. For example, PeopleSoft 7.56 is 7.x and PeopleSoft 8.16 is 8.1x.

Security Parameters

Security Enabled?

Determines whether to enforce PeopleSoft data access security. The options are *Yes* and *No*. *Yes* is the default value.

Specify Security Access By

Determines the way in which the PeopleSoft administrator enables users to access the server. The options are:

- **Class/Permission List.** Manages users in operating class (permission list) grouping.
- **User/Operation ID.** Manages users by individual user IDs.

Connection Parameters

Database Type

Is the Data source type. The options are: Oracle, MS SQL Server, and DB2 Universal Database.

Database Server

Is the name of the Database Server (for SQL Server only, not Oracle or DB2).

Database Identifier

Is the data source identifier or system data source name.

If the server is co-located with the data residing in Oracle, do not enter anything in the Database Identifier field.

Access ID

Is the RDBMS login ID for data source connectivity. Typically, it is the same login ID PeopleSoft uses to access the data source. There are minimum data access requirements that the ID must have.

Access Password

Is a password associated with the access ID.

Database Owner

Is the data source owner identifier or the schema owner depending on your data source. The owner ID is used to fully qualify SQL requests passed to the data source. This allows multiple data source instances or schemas to exist in a single data source. Typical defaults are SYSADM on Oracle and dbo on Microsoft SQL Server.

Authentication Options

Authentication

Determines the authentication used in your application. The options are:

- **TRUSTED (Alternate Authentication).** Does not prompt for security credentials. This can be used in applications that reside in a secure environment and where the user has been previously authenticated and. This kind of authentication requires that a valid PeopleSoft user ID be passed to the server to enable data access.
- **Password Required (Standard).** Prompts a user for a user ID and password before allowing them to view data. This is the default and is recommended for most environments.

Enforce Mixed Case IDs

Determines whether passwords are case-sensitive. Select *Yes* to enforce the PeopleSoft 8 use of mixed-case user IDs or select *No* to revert to PeopleSoft 7.x case insensitivity.

Application Server

Is the PeopleSoft application server name or IP address to be used for authentication.

JOLT Listener Port

Is the active Jolt listener port number to be used for authentication. The default in most environments is 9000.

Synonym Options

Default Synonym

Is the type of synonym to default to when creating new metadata. The options are: Record Name, Record Name with Prefix, Record Name with Suffix, Table Name, Table Name with Prefix, Table Name with Suffix.

Prefix, Suffix Default

Is the default prefix or suffix value if the default synonym setting requires it. This value is ignored if you use Record Name or Table Name as the default synonym.

Adding Additional PeopleSoft Connections

After you add your first connection, adding additional PeopleSoft connections is slightly different. The difference is that prior to managing metadata and connection information, you are not prompted for the DBA.

Note: All additional PeopleSoft data sources must utilize the same DBA in order for base metadata to function properly.

Procedure How to Add a New PeopleSoft Data Source Instance

1. From the Web Console, click *Connections* under the Administration group.

The Administration - Maintain Connections window opens:

The screenshot shows the 'Administration - Maintain Connections' window. On the left, there is a navigation menu with the following items: **Synonyms** (Create, Remove, Re-synch, Refresh, View, Edit), **Security** (Add Access, Remove Access, Re-Synch), and **Administration** (Reports, Connections, DBA Password, Set Tracing). The main content area is titled 'Administration - Maintain Connections'. It features a 'Select Connection:' dropdown menu with 'PS HRMS 8sp3' selected. Below this, there are three radio buttons: 'Update Existing' (selected), 'Add New', and 'Remove Existing'. Another 'Select PeopleSoft Connection:' dropdown menu also has 'PS HRMS 8sp3' selected. At the bottom of the main area, there are two buttons: 'Next >' and 'Reset'.

2. Select *Add New*.
3. Click *Next>*.

The Administration - Create a New PeopleSoft Connection window opens. For details, see *Administration - Create a New PeopleSoft Connection Window* on page 28-19.

When processing finishes, you can manage your synonyms and security for this connection.

Removing a PeopleSoft Connection

The integration with PeopleSoft depends on valid connection information. If a previously created connection is no longer required, you should remove that connection.

Procedure How to Remove a PeopleSoft Connection

1. From the Web Console, click *Connections* under the Administration group.

The Administration - Maintain Connections window opens:

2. Select a connection from the Select PeopleSoft Connection drop-down list.
3. Select *Remove Existing*.
4. Click *Next>*.

The Administration - Remove a PeopleSoft Connection window opens up.

You are prompted to confirm. After confirmation, the system removes the selected PeopleSoft Connection and all of its associated metadata.

Updating a PeopleSoft Connection

If it becomes necessary to update the connection parameters to a PeopleSoft data source, this can be accomplished through the Administration options.

The following functions may be performed:

- Turn PeopleSoft data access security enforcement on or off.
- Toggle between trusted and password authentication.
- Manage the user access specification type.
- Manage all data source identification Information.
- Define default synonym behavior.

Procedure How to Edit a PeopleSoft Connection

1. From the Web Console, click *Connections* under the Administration group.

The Administration - Maintain Connections window opens:

The screenshot shows the 'Administration - Maintain Connections' window. On the left, there is a navigation menu with the following items:

- Synonyms**
 - > Create
 - > Remove
 - > Re-synch
 - > Refresh
 - > View
 - > Edit
- Security**
 - > Add Access
 - > Remove Access
 - > Re-Synch
- Administration**
 - > Reports
 - > Connections
 - > DBA Password
 - > Set Tracing

The main content area is titled 'Administration - Maintain Connections'. It contains the following elements:

- 'Select Connection:' dropdown menu with 'PS HRMS 8sp3' selected.
- 'What to do?' section with three radio buttons:
 - Update Existing
 - Add New
 - Remove Existing
- 'Select PeopleSoft Connection:' dropdown menu with 'PS HRMS 8sp3' selected.
- 'Next >' and 'Reset' buttons.

2. Select a connection from the Select PeopleSoft Connection drop-down list.

3. Select *Update Existing*.

4. Click *Next>*.

The Administration - Update a PeopleSoft Connection window opens. For details, see *Administration - Create a New PeopleSoft Connection Window* on page 28-19.

5. Make the modifications you wish, and click *Next>*.

Reference Updating Connections

- When changing the Specify Security Access By setting, security must be resynchronized before changes will take place. It is recommended that all currently configured users be removed and added again after security settings have been modified.
- When changing authentication settings, ensure to consider the logical process order. For example, to gain access to the Configuration Update utility, you must first log in. The authentication information can be modified through the Data Adapter for PeopleSoft and stored. After that is accomplished, the actual PeopleSoft API settings can be modified.

As when modifying the authentication information, care must be taken to manage changes to data source connectivity correctly. Always modify your settings in the PeopleSoft Connection first. Then, change them at the data source. Where possible, keep both access options in place during a configuration conversion process until tests have proven a successful switch.

Using the PeopleSoft Metadata and Security Administrator

This topic describes how to use the PeopleSoft Metadata and Security Administrator. The PeopleSoft Metadata and Security Administrator is a graphical interface used to perform administrative functions through the Web Console.

The Administrator enables you to:

- Manage PeopleSoft record metadata.
- Manage security access.
- View server settings.

Note: The procedures in this topic show windows in a security environment based on user IDs. If your security is based on permission lists, some screens may appear slightly different in your application.

Accessing the PeopleSoft Metadata and Security Administrator

To access the environment for administration, you must have access to the Web Console and knowledge of the DBA.

Procedure How to Access the PeopleSoft Metadata and Security Administrator

1. Open the Web Console.

You can access the Web Console through a Web browser by opening the URL to the server's HTTP listener port. The default location on a local server is `http://localhost:8121/`.

Note: If the Web Console is set to Protected mode, you must supply a Web Console administrator ID and password to gain access.

2. Select *Data Adapters* in the main window.

The Configuring Data Adapters window opens.

3. Select *PeopleSoft* in the Configured folder, and select *Properties* from the pop-up menu.

The Administration - Enter DBA Password window opens.

4. Enter the DBA password, and click *Next*>.

The PeopleSoft menu opens:



Managing PeopleSoft Metadata

The PeopleSoft Metadata and Security Administrator enables management of PeopleSoft metadata. Processes exist that manage the accessibility of PeopleSoft records by maintaining the metadata catalog on the reporting server. Before reports can be created or run, this metadata catalog must contain information about the PeopleSoft record definitions. You can perform with following tasks to accomplish this:

- Create synonyms for PeopleSoft records.
- Remove synonyms for PeopleSoft records.
- Resynchronize synonyms.

Creating Synonyms

Creating synonyms for PeopleSoft is one of the core functions of the PeopleSoft Administrator. The adapter enables the administrator to choose which PeopleSoft records are defined in the metadata catalog.

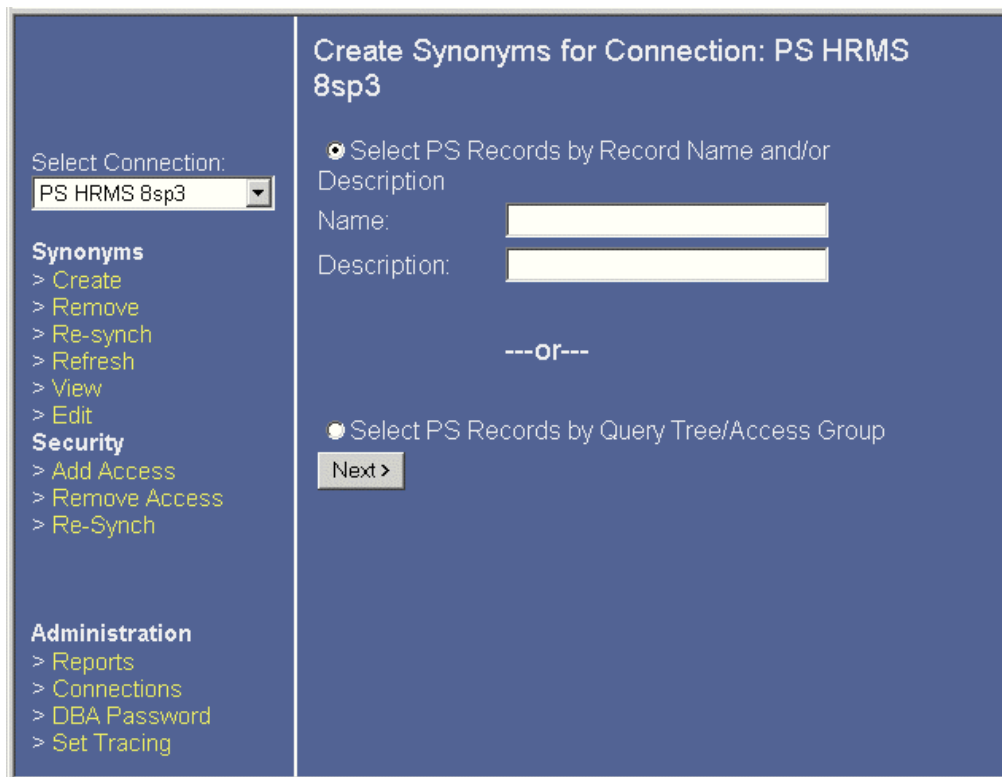
The data adapter provides two methods to select PeopleSoft records to create synonyms:

- **Create synonyms by record definition search template.** A search template option is provided to speed the process of searching for specific PeopleSoft records. This is a useful method of locating record definitions when either the query tree location is not known or specific definitions that meet a filter can be more readily accessed.
- **Create synonyms with the query tree search.** The query tree is a hierarchical logical grouping of the PeopleSoft record definitions. This method provides administrators familiar with this organizational structure a way to quickly locate related record definitions. Many sites create a combination query tree and access group definition specifically to group their records for use in reporting.

Procedure **How to Create Synonyms Using the Record Definition Search**

1. In the PeopleSoft menu, select the connection you want to manage from the Select Connection drop-down list. For details on accessing the PeopleSoft menu, see *How to Access the PeopleSoft Metadata and Security Administrator* on page 28-26.
2. Click *Create* under the Synonyms group.

The Create Synonyms for Connection window opens:



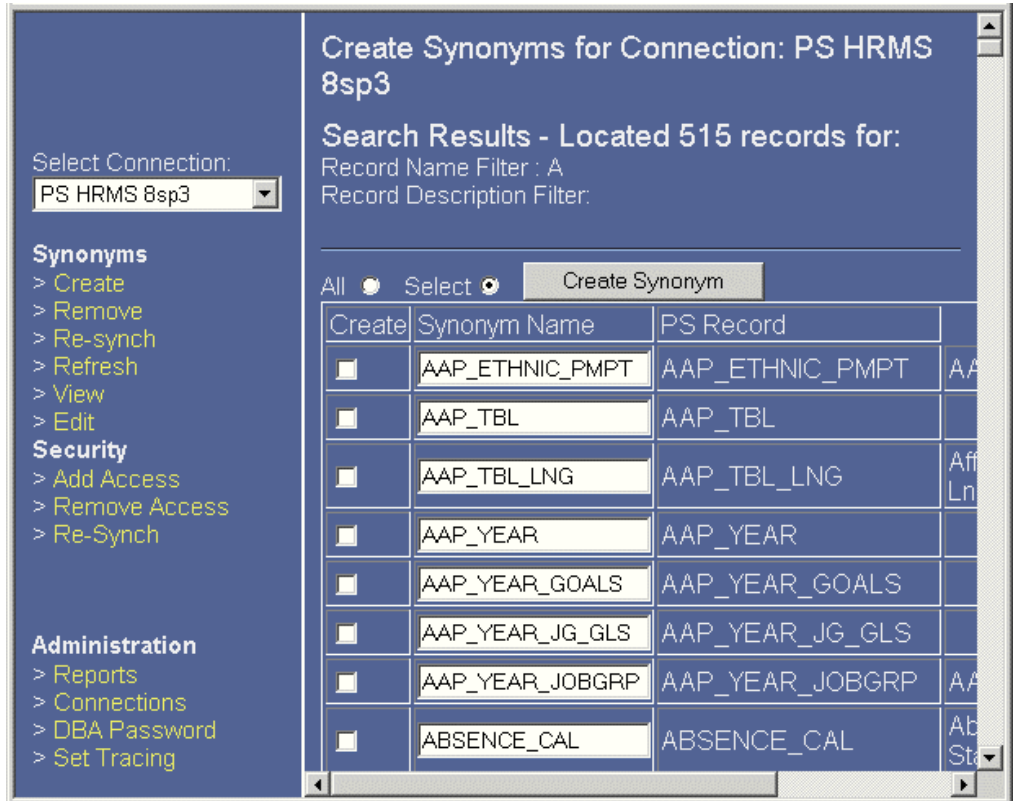
3. Click *Select PS Records by Record Name and/or Description* and enter a record name or description to filter by in the available fields, or leave the fields blank to see a full list of records.

Note:

- The record description is case sensitive.
- If you do not use a filter, or your filtered results are very large, the results may be truncated. Use a different filter to reduce the returned results.

4. Click *Next*>.

The Create Synonyms for Connection window opens:



5. Select the check box in the Create column for all the records you want to create a synonym for, or select *All* to create a synonym for all available records.
6. Optionally, change the synonym name in the Synonym name column.
7. Click *Create Synonym* to process. Processing may take a few minutes for each selected record definition.

Procedure **How to Create Synonyms Using Query Tree Search**

1. In the PeopleSoft menu, select the connection you want to manage from the Select Connection drop-down list. For details on accessing the PeopleSoft menu, see *How to Access the PeopleSoft Metadata and Security Administrator* on page 28-26.
2. Click *Create* under the Synonyms group.

The Create Synonyms for Connection window opens.

Select Connection:
PS HRMS 8sp3

Synonyms
> Create
> Remove
> Re-synch
> Refresh
> View
> Edit

Security
> Add Access
> Remove Access
> Re-Synch

Administration
> Reports
> Connections
> DBA Password
> Set Tracing

Create Synonyms for Connection: PS HRMS 8sp3

Select PS Records by Record Name and/or Description
Name:
Description:

---or---

Select PS Records by Query Tree/Access Group

Next >

3. Click *Select PS Records by Query Tree/Access Group* and click *Next>*.

The Create Synonyms - Select Query Tree for Connection window opens:

Select Connection:
PS HRMS 8sp3

Synonyms
> Create
> Remove
> Re-synch
> Refresh
> View
> Edit

Security
> Add Access
> Remove Access
> Re-Synch

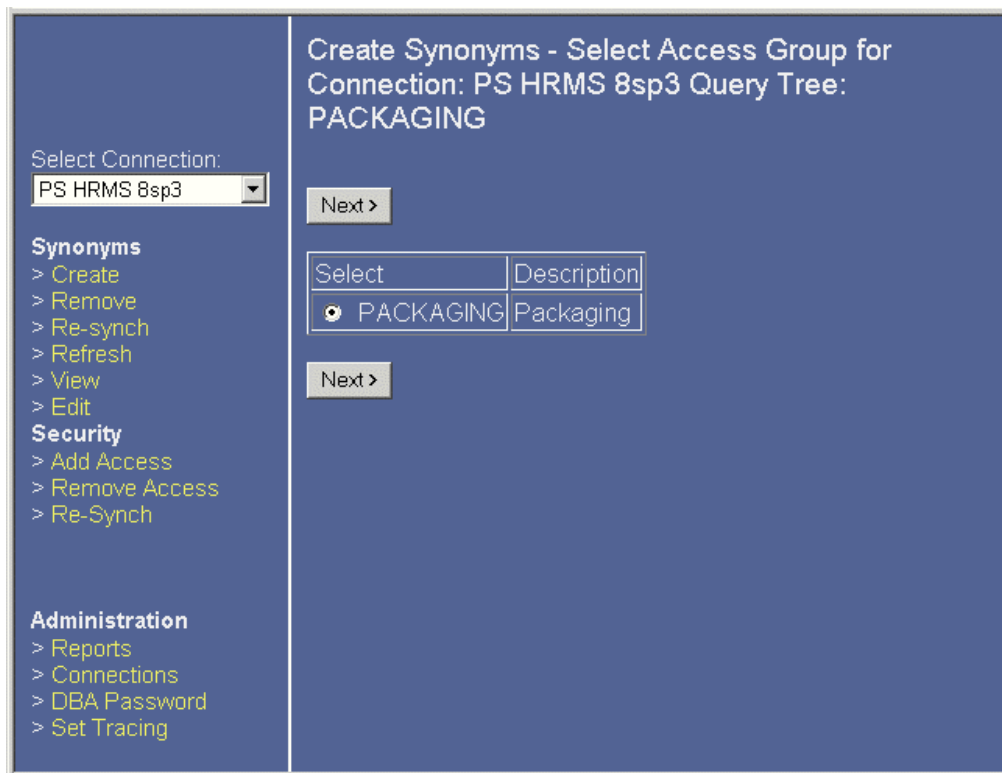
Administration
> Reports
> Connections
> DBA Password
> Set Tracing

Create Synonyms - Select Query Tree for
Connection: PS HRMS 8sp3
Next >

Select	Description
<input checked="" type="radio"/> PACKAGING	EM Product License Packaging
<input type="radio"/> QUERY_TREE_BAS	Benefits Admin Access Group
<input type="radio"/> QUERY_TREE_BENEFIT	Benefits Access Group
<input type="radio"/> QUERY_TREE_EO	Enterprise Components
<input type="radio"/> QUERY_TREE_FSA	Flex Spending Admin Access Grp
<input type="radio"/> QUERY_TREE_GP	Global Payroll
<input type="radio"/> QUERY_TREE_GPAUS	Global Payroll AUS Query Tree
<input type="radio"/> QUERY_TREE_GPCHE	Global Payroll CHE Query Tree
<input type="radio"/> QUERY_TREE_GPDEU	Global Payroll DEU Query Tree
<input type="radio"/> QUERY_TREE_GPESP	Global Payroll ESP Query Tree

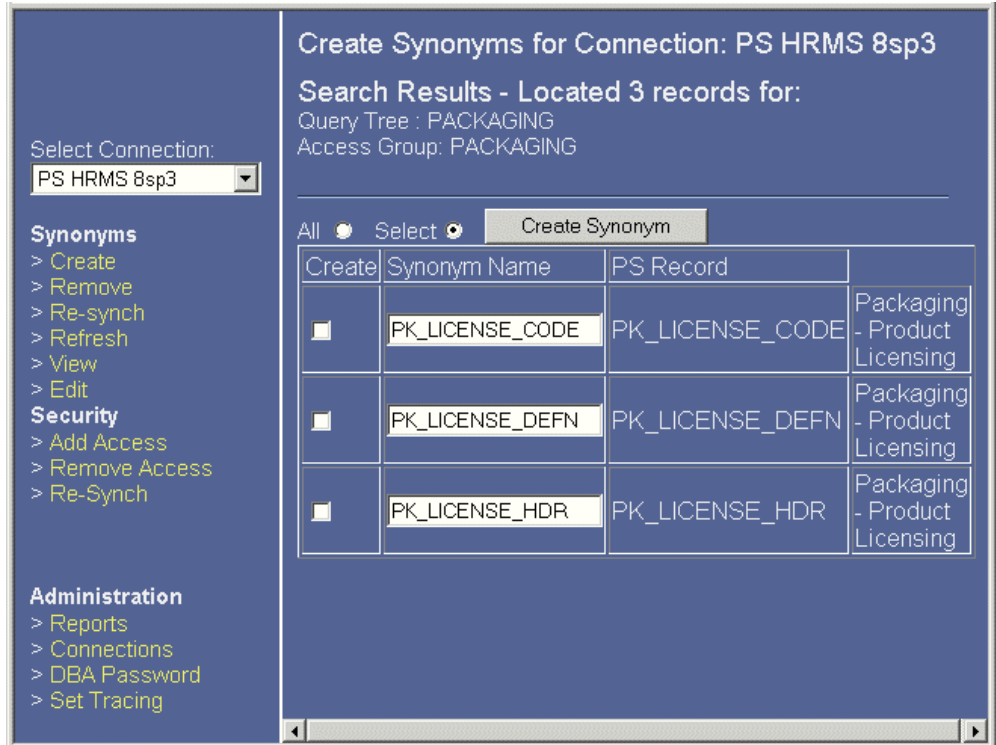
4. Select a query tree from the table, and click *Next*>.

The Create Synonyms - Select Access Group for Connection window opens:



5. Select an access group to drill-down on from the Select Column, and click *Next>*. If there is only one access group listed, click *Next>*.

The Create Synonyms for Connection window opens:



6. Select the check box in the Create column for all the records you want to create a synonym for, or select *All* to create a synonym for all available records.
7. Optionally, change the synonym name in the Synonym name column.
8. Click *Create Synonym* to process. Processing may take a few minutes for each selected record definition.

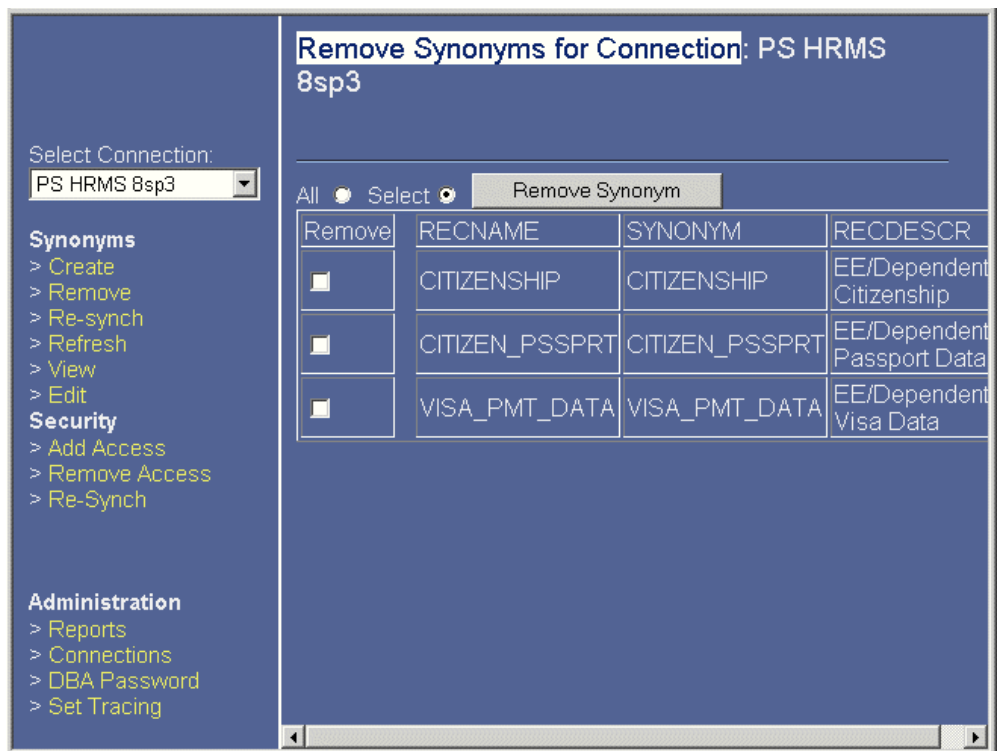
Removing Synonyms

Previously created PeopleSoft synonyms can be removed from the metadata catalog with the Remove Synonyms menu option.

Procedure How to Remove Records

1. In the PeopleSoft menu, select the connection you want to manage from the Select Connection drop-down list. For details on accessing the PeopleSoft menu, see *How to Access the PeopleSoft Metadata and Security Administrator* on page 28-26.
2. Click *Remove* under the Synonyms group.

The Remove Synonyms for Connection window opens:



3. Select the check box in the Remove column for each synonym you want to remove, or select *All* to remove all available synonyms.
4. Click *Remove Synonym*.

Resynchronizing Synonyms

You can resynchronize your metadata. This option:

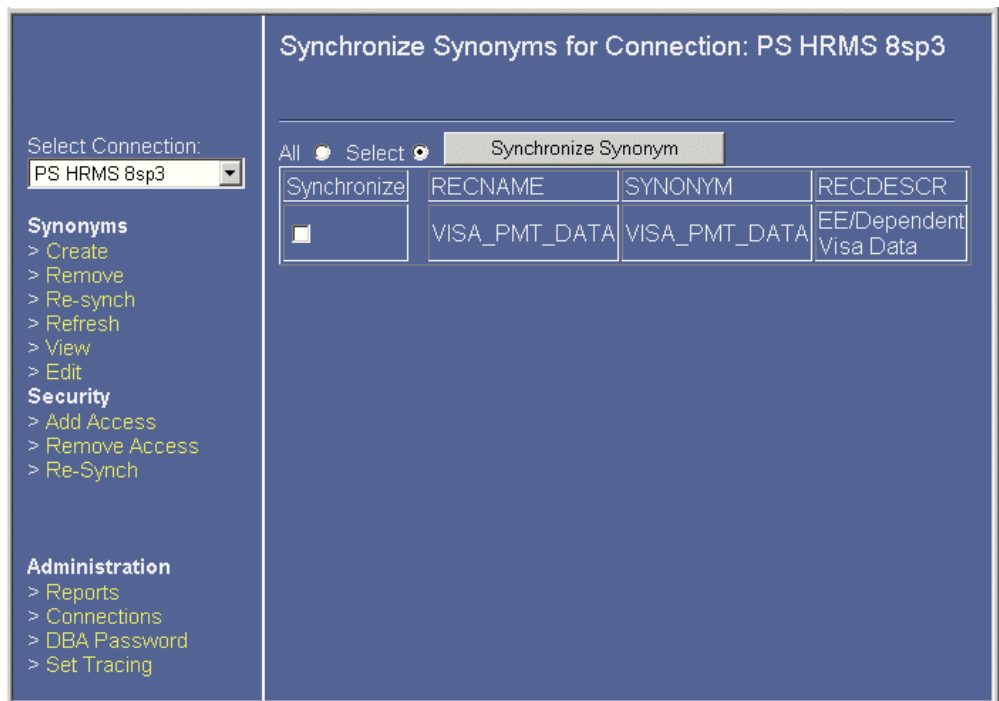
- Queries the internal repository to determine which records were installed for the selected PeopleSoft connection.
- Compares the version number in those records with the version number in the PeopleSoft data source. If none of the records have changed, a message appears.

If there are any records that have changed, they appear in the Select Record(s) to Synchronize page.

Procedure How to Resynchronize Metadata

1. In the PeopleSoft menu, select the connection you want to manage from the Select Connection drop-down list. For details on accessing the PeopleSoft menu, see *How to Access the PeopleSoft Metadata and Security Administrator* on page 28-26.
2. Click *Re-Synch* under the Synonyms group.

The Synchronize Synonyms for Connection window opens:



3. Select the check box in the Synchronize column for each synonym you want to resynchronize with PeopleSoft metadata definitions, or select *All* to resynchronize all listed synonyms.
4. Click *Synchronize Synonym*.

Security Access Administration

Before a PeopleSoft user can run reports against PeopleSoft, security access rights must be granted. The PeopleSoft connection does not automatically enable reporting for every user; instead an administrator must decide which users have reporting access.

An administrator can manage security access according to individual users or a permission list. The decision for how to manage access is made as a connection option, and the two methods are mutually exclusive.

Regardless of the method for granting access rights to users, the process for managing security access is roughly identical. The administrator can affect security access for a user in the following ways:

- Adding security access.
- Removing security access.
- Resynchronizing security access.

Adding Security Access

Before a PeopleSoft user can run reports against PeopleSoft, security access rights must be granted. The PeopleSoft connection does not automatically enable reporting for every user; instead an administrator must decide which users have reporting access.

Procedure How to Add Security Access for a User

1. In the PeopleSoft menu, select the connection you want to manage from the Select Connection drop-down list. For details on accessing the PeopleSoft menu, see *How to Access the PeopleSoft Metadata and Security Administrator* on page 28-26.
2. Click *Add Access* under the Security group.

The Add Security Access to Connection window opens:

Select Connection:
PS HRMS 8sp3

Synonyms
> Create
> Remove
> Re-synch
> Refresh
> View
> Edit

Security
> Add Access
> Remove Access
> Re-Synch

Administration
> Reports
> Connections
> DBA Password
> Set Tracing

Add Security Access to Connection: PS HRMS 8sp3

Select Users:

Name:

Description:

Next >

3. Select filtering criteria to reduce the list of users, and click *Next>*, or leave the fields blank and click *Next>* to see a full list of users.
 - If your connection is set up to manage security by user ID, then provide user name or description filtering criteria.
 - If your connection is set up to manage security by permission list, then provide relevant permission list criteria.

Note: If you do not use a filter, or your filtered results are very large, the results may be truncated. Use a different filter to reduce the returned results.

The Add Security Access to Connection window opens:



4. Select the check boxes in the Add column for the user IDs or permission lists you would like to add access for, or select *All* to add access for all available users.
5. Click *Add Users* to add the selected users.

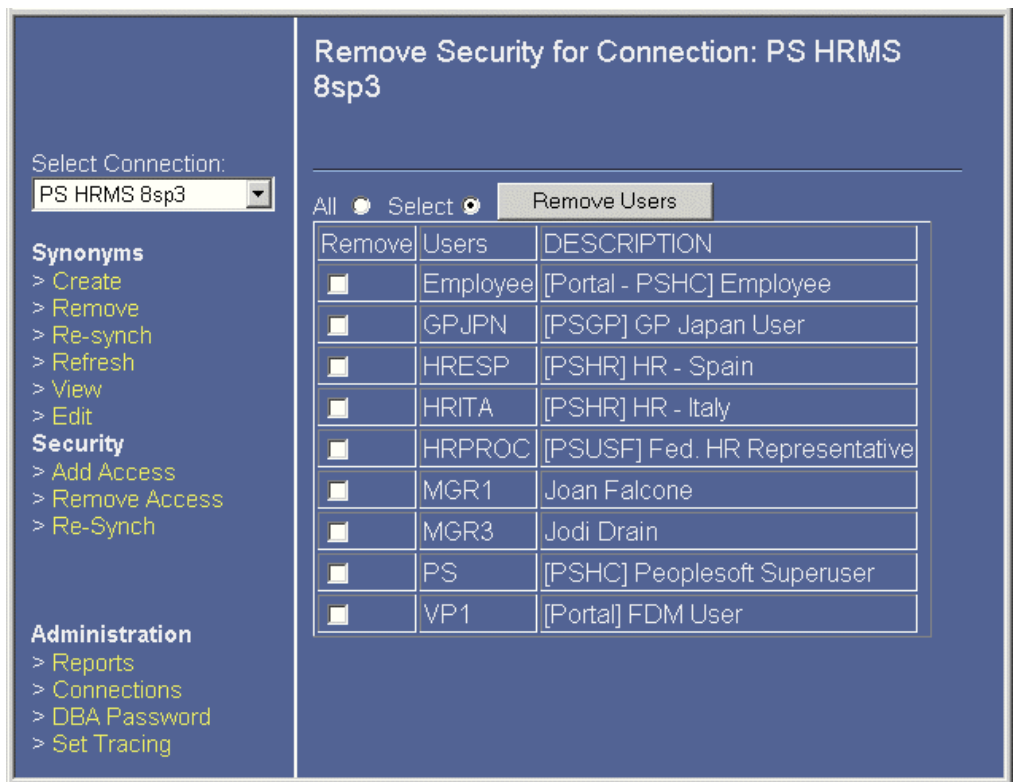
Removing Security Access

You can remove security access for a user from the PeopleSoft menu.

Procedure How to Remove Security Access for Users

1. In the PeopleSoft menu, select the connection you want to manage from the Select Connection drop-down list. For details on accessing the PeopleSoft menu, see *How to Access the PeopleSoft Metadata and Security Administrator* on page 28-26.
2. Click *Remove Access* under the Security group.

The Remove Security to Connection window opens:



3. Select the check boxes in the Remove column for the user IDs or permission lists that should be removed, or select *All* to remove access for all available users.
4. Click *Remove Users*.

The PeopleSoft connection removes all references to these users or permission lists. After they are removed, the users or classes cannot access PeopleSoft data through PeopleSoft.

Resynchronizing Security

You can resynchronize PeopleSoft security metadata with changes in the PeopleSoft environment. The following situations require resynchronization:

- A permission list has had query access group privileges modified.
- A record has been added or removed from a query tree.
- A user has been added to or removed from a permission list with security access enabled. This can occur directly, as in PeopleSoft 7.x, or indirectly through role assignment changes.
- Row level security class or permission lists are changed.

You do not need to run the security resynchronization routine if row-level security access has changed, based on the security table being updated. This type of security change will be enforced automatically and immediately. If the security search record within a record definition has been changed, the record metadata resynchronization routine must be run.

Note: A batch mechanism for resynchronization is also available that may be scheduled through a command line execution. For more information, see *Advanced Topics* on page 28-42.

Procedure How to Resynchronize Security Rules

1. In the PeopleSoft menu, select the connection you want to manage from the Select Connection drop-down list. For details on accessing the PeopleSoft menu, see *How to Access the PeopleSoft Metadata and Security Administrator* on page 28-26.
2. Click *Re-Synch* under the Security group.

Security rules are resynchronized based upon security access settings. Processing may take several minutes.

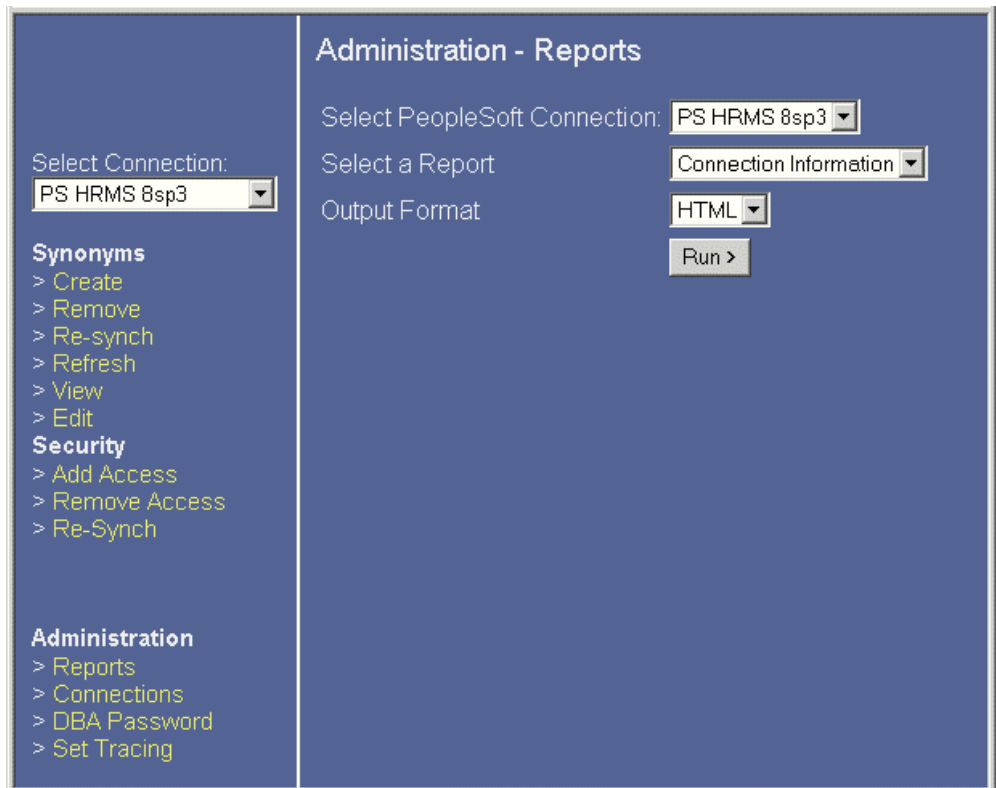
Administration Reports

The Administrator provides reports to make administration easier. You can run reports that provide information on connections, metadata, users, and user access. These reports are accessible from the index.

Procedure How to Run Administration Reports

1. In the PeopleSoft menu, select the connection you want to manage from the Select Connection drop-down list. For details on accessing the PeopleSoft menu, see *How to Access the PeopleSoft Metadata and Security Administrator* on page 28-26.
2. Click *Reports* under the Administration group.

The Administration - Reports window opens:



3. If necessary, select the PeopleSoft connection you want to run a report for from the Select PeopleSoft Connection drop-down list.

4. Select a report type from the Select a Report drop-down list. The options are:
 - **Connection Information.** Retrieves statistics for your connection.
 - **Synonyms.** Retrieves a list of available synonyms.
 - **Users/Oprids.** Retrieves a list of users who have access to the current connection.
 - **Access Assigned.** Retrieves a list of the access assigned to each user.
5. Select an output format for the report from the Output Format drop-down list. The options are *HTML* and *PDF*.
6. Click *Run>*.

The selected report appears.

Advanced Topics

This topic discusses advanced usage techniques in the PeopleSoft environment. These topics describe configuration, additional functionality, and integration problems.

Stand-alone Server Usage

A server can be accessed locally without the use of client software. This functionality can be started in the following two modes:

- **Interactive Agent** is useful for:
 - Debugging application errors.
 - Troubleshooting connectivity and configuration.
 - Testing performance.
- **Batch Mode** is useful for:
 - Scheduling batch scripts with operating system tools.

An administrative user with access to the server can use the server in stand-alone mode to perform any of these functions. If PeopleSoft data must be accessed in any of these processes, you must execute the PeopleSoft authentication routine. Because of the processing within the authentication routine, the creation of a custom startup script is recommended.

The two purposes for the custom script are:

- To enable successful execution of the PSLOGIN remote procedure call.
- To ensure that when executing, a separate and temporary workspace is used. This prevents errors due to concurrent processing, including data access errors.

When using the edastart script, temporary files are always written to the current directory. When using edastart without modification, be sure to delete these temporary files from the current directory after execution to avoid unexpected results during run time.

Example **Creating a Custom Server Startup Script for UNIX**

The following provides the steps for creating and using a custom startup script in UNIX. A similar script can be created in Windows with appropriate operating system-specific syntax.

Navigate to the EDACONF/bin directory on your server and create a new script with a text editor. The following script is named pslocal. The numbers on the left correspond to the notes explaining the code.

Note: This example creates a unique directory under the \$EDACONF/EDALOCA directory. The EDALOCA directory must exist in most operating systems before a directory can be created within it.

```

#!/bin/ksh
#
parms=$*
#
# Define the Conf Directory
1. export EDACONF=/ibi/srv52/wfs
#
# Create unique temporary directory for running locally
2. EDA_PID=$$
   export EDATEMP=$EDACONF/edalocal/$EDA_PID
   rm -rf $EDATEMP
   mkdir $EDATEMP
3. cd $EDATEMP
4. $EDACONF/bin/edastart $parms
5. cd $EDACONF/bin
   rm -r $EDATEMP

```

The script processes as follows:

1. The export command defines the EDACONF variable to use. This makes the remainder of the script more flexible and portable.
2. These four lines define a unique directory based on a PID and UNIX process ID so that all temporary files will be written to a unique location.
3. The CD command changes the current directory to the new temporary directory.
4. This line calls the delivered edastart script and parameters passed through the script.
5. These two lines change the current directory from the temporary space, and deletes the temporary files.

Procedure How to Use the PeopleSoft Interactive Agent

1. Navigate to the server configuration directory, then to the bin subdirectory.
2. Execute the Interactive Agent start-up script with the -t option. This is the same for UNIX and Windows platforms.

```
/ibi/srv52/wfs/bin/pslocal -t
```

where:

```
/ibi/srv52/wfs/bin
```

Is the bin directory under the server configuration.

When you run

```
pslocal -t
```

you see information similar to:

```
RELEASE = R729520B  
GEN_NUM = 001.123456  
SOURCE_DATE = 01/01/2003 19:36:53  
BUILD_DATE = 01/01/2003 02:01:09  
INSTALLATION_DATE= 01/01/2003 02:46:42  
>>
```

At the prompt, you can enter commands.

3. To perform any commands that access PeopleSoft data, first you must run the interactive authentication routine. If this procedure is not run, you receive data access errors. To execute the procedure, type

```
>>EX PSLOGIN USERID=PS_Userid, PASSWD=PS_Password
```

where:

```
PS_Userid and PS_Password
```

Is a valid PeopleSoft user ID and password.

You see processing information followed by a FOCUS command prompt. After control returns to the FOCUS command prompt, PeopleSoft data may be accessed with the FOCUS language.

Note: If your configuration has been set up with "Trusted" authentication, then you are not required to supply the PeopleSoft password. Only a valid user ID is required.

Batch Mode Security Resynchronization

Security resynchronization is one of the functions of the PeopleSoft Administrator. When administering PeopleSoft connections through the Web console, you simply select the Re-Synch link under the Security group, and all of the PeopleSoft data security rules are resynchronized.

The batch resynchronization utility enables administrators to schedule, through an operating system command, the execution of this functionality without having to manually open the Administrator tool. By scheduling this process nightly (or otherwise; based on business requirements), administrators are not required to perform this function manually, and they can be assured that the data security rules are current as of the last scheduled run.

The utility is comprised of two files. These files are a procedure (pssecsnk.fex) used in executing the resynchronization routine and a t3i script (pssecsnk.t3i), which is the command line input file. By starting the server with the t3i script as an input, the server performs the required security resynchronization without user intervention.

Procedure How to Run Batch Mode Security Resynchronization From the Command Line

1. Locate the pssecsnk.t3i script in the EDACONF\catalog directory.
2. Using PeopleSoft Batch-Mode, execute with the security synchronization input script. To execute input scripts with the *pslocal* script, use the -f option

```
c:\ibi\srv52\wfs\bin\pslocal -f c:\ibi\srv52\wfs\catalog\pssecsnk.t3i
```

where:

```
c:\ibi\srv52\wfs\catalog
```

Is the directory location of the t3i file.

Copy it here from the installation location under c:\ibi\srv52\home\catalog.

Password Security

The t3i script is an ASCII text file that must be edited for your particular environment. Inside are two FOCUS execute command lines that:

- Log into PeopleSoft.
- Perform the security resynchronization.

The following is the exact code:

```
< Filename: pssecsnk.t3i >  
%CONNECT  
%BEGIN  
EX PSLOGIN USERID=PS, PASSWD=PS  
EX PSSECBAT  
%END  
%DISCONNECT  
%STOP_SERVER
```

You must modify the user ID and password in accordance with one that is appropriate for your environment. The user ID must be granted access in PeopleSoft.

Since the file is in human readable text, the administrator must prevent read/write access for anyone not authorized to access the t3i. The best way to prevent unauthorized access is to use operating system security. Windows and UNIX provide ways to prevent unauthorized access to individual files and entire directories. Contact your operating system administrator for assistance in performing this security step.

Execution Results

Usually, after execution all temporary work files are deleted immediately. If you must verify work files or see the batch process output file, you can temporarily modify the execution script.

To temporarily prevent the deletion of these processing files, edit the pslocal script found in the EDACONF/bin directory. Insert the appropriate comment character for your environment where the final file deletion step exists.

In UNIX environments:

```
# rm -r $EDATEMP
```

After the script has been executed, the Administrator can review the execution results. A batch process output file (pssecsnk.t3o) is automatically created in the EDACONF/edalocal directory, with a unique subdirectory for each execution. The following is an example of a completed execution that was successful:

```
< Filename: pssecsnk.t3o >
%connect
%begin
ex pslogin userid=ps, passwd=ps
ex pssecbat
%end
SYSPRINT.SCR
0 NUMBER OF RECORDS IN TABLE=      1  LINES=      1
File doesn't exist
  DBNUM: 1 has been re-synchronized, DBA security is: ON
CLOSEALL.ALL
%disconnect
%stop_server
```

Note: As the user ID and password appear in this output file, the file must be protected by the operating system security.

Additional Options

The following advanced options are available to simplify resynchronizing security in batch mode:

- **Process Scheduling.** Because the t3i script is run from an operating system command line, you can execute this procedure with any scheduling software on the server that accepts operating system syntax. For more information, see your operating system or process scheduling software documentation.
- **Scheduling with ReportCaster.** An alternative approach to using the t3i script for resynchronizing security is to use ReportCaster (if available at your site) to schedule the execution. The procedure that must be scheduled is PSSECBAT.FEX. It runs under the user ID used to schedule the execution.

The major benefit to using ReportCaster is that all administrators are not required to use operating system security to prevent unauthorized access to the t3i script. Additionally, it is easy to add pre-processing and post-processing steps. An example of a post-processing step is to run a procedure that sends an e-mail message to the administrator every time the resynchronizing occurs with a completion status.

Troubleshooting Tips

These troubleshooting tips help if you are getting FOCUS error messages, such as a FOC1302, or if you are seeing SQL errors when you run reports.

Procedure How to Connect to the PeopleSoft Data Sources

The most likely source of difficulty in connecting to your PeopleSoft data source is the data source connectivity. If you are experiencing difficulty connecting to the data source:

1. Verify that the data source instance is up and running.
2. Run a client query tool on the server, using the PeopleSoft access ID.
 - If you cannot log in to the data source, there is a problem with the access ID. Verify the ID and check to see if the password has changed.
If the password has changed, then it must also be changed in the PeopleSoft configuration.
 - If you can log in, proceed to Step 3.
3. Run a few simple queries against the PeopleSoft records that are returning errors.

If you are not getting any rows back, check to see if the PeopleSoft access ID has read authority on the PeopleSoft records you are accessing.

Procedure How to Verify Security

PeopleSoft enforces all the security mechanisms used by the PeopleSoft Query tool. As of PeopleSoft Release 7.5, this includes Query Tree security and row-level security. If you are getting unexpected results, it is possible that your PeopleSoft security has not been configured properly.

To verify that security is being properly enforced:

1. Verify that the synonym(s) being queried are correct.

The Administrator can run a report of installed records to verify. If row-level security is the issue, the server-side ACX file should be reviewed and compared to the record definition in PeopleSoft, including the Security Search record.

2. Create a similar query in PS/Query. The SQL code can be viewed and compared to SQL tracing on the server.

3. Examine the SQL.

Does it appear to match the business logic behind the original request? If not, check the syntax behind the original FOCUS code. Perhaps you are using a WRITE instead of a PRINT, or perhaps a WHERE clause has been coded incorrectly.

4. Run this SQL directly against the RDBMS using a data source query tool.

If you are getting the expected results, then there may be a problem with the program logic after the data is returned to FOCUS from the RDBMS. Examine COMPUTE/DEFINE statements for logic errors; or if you are using ON TABLE HOLD, there may be a problem in your code.

If you are getting the same unexpected results, then examine the underlying data. Perhaps some cross-referenced fields are missing, or perhaps row-level security is eliminating essential rows.

CHAPTER 29

Getting Started in Progress

Topics:

- Preparing the Progress Environment
- Configuring the Data Adapter for Progress
- Managing Progress Metadata
- Customizing the Progress Environment

The Data Adapter for Progress allows applications to access Progress data sources. The adapter converts application requests into native Progress statements and returns optimized answer sets to the requesting application.

Preparing the Progress Environment

Currently, the Progress environment is supported on UNIX and Windows NT/2000. The Progress Adapter is available as an ODBC interface on all platforms except Compaq/Alpha Tru64. On Compaq/Alpha Tru64 only the embedded SQL interface is available.

Procedure How to Set Up the Environment on Windows NT/2000 Using ODBC

On Windows NT/2000, the Progress environment is set up during the installation of the product.

Procedure How to Set Up the Environment on UNIX Using ODBC

1. Specify the full name of the directory containing Progress.

```
DLC = /rdbms/pro91d/dlc
```

2. Specify the full name of the default Progress startup file.

```
PROSTARTUP = /rdbms/pro91d/dlc/startupyy.pf
```

3. Specify the location of the ODBC initialization file.

```
ODBCINI = /usr/odbc/ibiodbc/progress.ini
```

4. Specify the path to Progress shared libraries.

```
LIBPATH=/rdbms/pro91d/dlc/odbc/lib : /rdbms/pro91d/dlc/lib
```

Procedure How to Set Up the Environment on Compaq/Alpha Tru64 (non-ODBC)

1. Specify the location where Progress is installed.

```
PRO_HM = /rdbms/pro91d/qaeda
```

2. Specify the file that contains the Progress text messages.

```
PROMSGS = /rdbms/pro91d/dlc/promsgs
```

3. Specify the full name of the directory containing Progress.

```
DLC = /rdbms/pro91d/dlc
```

4. Specify the full name of the directory where Progress PROBUILD is installed.

```
PROLOAD = /rdbms/pro91d/dlc/probuild
```

5. Specify the full name of the default Progress startup file.

```
PROSTARTUP = /rdbms/pro91d/dlc/startupyy
```

Configuring the Data Adapter for Progress

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Connecting to a Progress Database Server

For an ODBC interface, the server must be brought up in TCP/IP mode. On Compaq/Alpha Tru64 platforms you may bring up the server in either shared memory or TCP/IP mode.

Shared memory is an area in system memory that multiple users can access concurrently. Windows and most UNIX systems use shared memory. The version number must be the same for all processes that access shared memory configurations. Both the client and server must be running the same release and version number.

Using TCP/IP, you can run different versions of the client and server. For example, if the database version is 8, the client can be version 9. If the database is version 9, a version 8 client can not communicate with it. Clients are able to connect to database servers using TCP/IP within the same version and one version prior. Once the database is brought up using TCP/IP (when the server is started using -H, -S, and -N parameters), the FOCPF global variable must be set under the server.

Example Setting the FOCPF Global Variable

```
Export FOCPF=/ul/home/tcp.pf
```

The file tcp.pf:

```
-S sssss -H mmmm -N TCP
-S service name
-H host name
-N network type
```

These parameters must match the parameters used when you connected to the database using TCP/IP.

Declaring Connection Attributes

In order to connect to an Progress database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Manually add the command in the global server profile (edasprof.prf) or in a user profile (user.prf).

- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the global server profile. For more information, see *Declaring Progress Connection Attributes From the Web Console*.

You can declare connections to more than one Progress database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to Progress Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Syntax

How to Declare Connection Attributes Manually

```
ENGINE [SQLPRO] SET CONNECTION_ATTRIBUTES [data source]/user_ID,password
```

where:

SQLPRO

Indicates the Data Adapter for Progress. You can omit this value if you previously issued the SET SQLENGINE command.

data source

Is the name of the Progress data source you wish to access.

user_ID

Is the primary authorization ID by which you are known to Progress.

password

Is the password associated with the primary authorization ID.

The SET command also allows you to access a Progress database using the embedded SQL interface:

```
ENGINE SQLPRO SET DATABASE qaeda
ENGINE SQLPRO SET HOME /rdbms/pro91d/qaeda
ENGINE SQLPRO SET USER edaqa
ENGINE SQLPRO SET PASSWORD qaeda3x
```

Example Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to the Progress database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLPRO SET CONNECTION_ATTRIBUTES SAMPLESERVER/MYUSER,PASS
```

Reference Declaring Progress Connection Attributes From the Web Console

Field	Description
Datasource	Enter the Progress data source.
User	Enter a valid user for Progress.
Password	Enter a valid password for Progress.

The Progress non-ODBC adapter configuration screen displays the following fields:

Field	Description
Database name	Enter the Progress database name.
Home directory	Enter the full path to the database location.
User	Enter a valid user for Progress.
Password	Enter a valid password for Progress.

Overriding the Default Connection

Once connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax How to Change the Default Connection

```
ENGINE [SQLPRO] SET DEFAULT_CONNECTION [connection]
```

where:

SQLPRO

Indicates the Data Adapter for Progress. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the service name defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following error message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following error message is issued: FOC1671, Command out of sequence.

Example Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the Progress database server named SAMPLENAME as the default Progress database server:

```
ENGINE SQLPRO SET DEFAULT_CONNECTION SAMPLENAME
```


Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax **How to Control the Connection Scope**

```
ENGINE [SQLPRO] SET AUTODISCONNECT ON {FIN|COMMAND}
```

where:

SQLPRO

Indicates the Data Adapter for Progress. You can omit this value if you previously issued the SET SQLENGINE command.

FIN

Disconnects automatically only after the session has been terminated. This value is the default.

COMMAND

Disconnects automatically after each request. Depending on how often the event occurs, the AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server; it is related to the operating system and the data source.

Managing Progress Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Progress data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Progress table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Editing Metadata page opens.

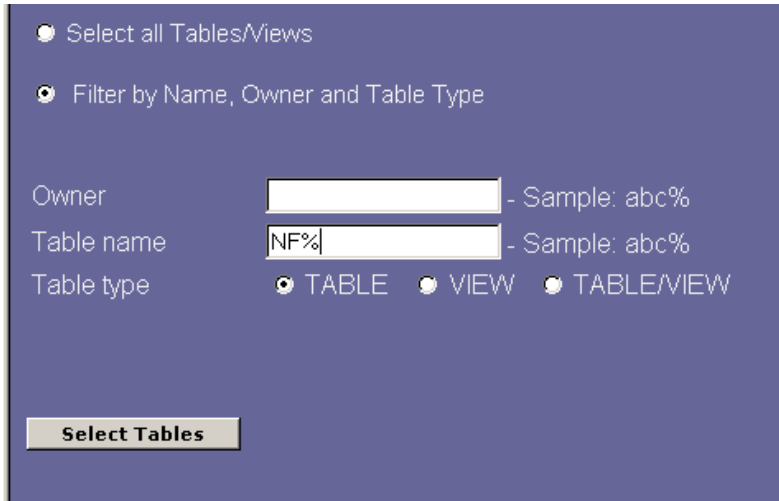
To create a synonym, you must have configured the adapter. See *Configuring the Data Adapter for Progress* for more information.

2. Expand the *Add* folder, expand the adapter folder, and then click a connection. The right pane displays table and view selection options:

Select All Tables/Views. Select this radio button to create synonyms for all tables and views. This value is the default.

Filter by Name, Owner and Table Type. Select this radio button to filter the tables or views for which to create synonyms.

Selecting this option adds the following:



The screenshot shows a dark blue panel with two radio button options. The first option, 'Select all Tables/Views', is unselected. The second option, 'Filter by Name, Owner and Table Type', is selected. Below the second option are three input fields: 'Owner' with a sample value 'abc%', 'Table name' with a sample value 'abc%' and the text 'NF%' entered, and 'Table type' with three radio buttons: 'TABLE' (selected), 'VIEW', and 'TABLE/VIEW'. At the bottom of the panel is a button labeled 'Select Tables'.

Owner. Enter a string for filtering the owners' IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owners' IDs begin with the letters ABC; %ABC to select tables or views whose owners' IDs end with the letters ABC; %ABC% to select tables or views whose owners' IDs contain the letters ABC at the beginning, middle, or end.

Table Name. Enter a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

Table Type. Select one of the following options: Tables (this is the default), Views, Tables/Views.

3. Click *Select Tables*. All tables that meet the specified criteria are displayed:

Prefix: SET CONVERSION LONGCHAR: TEXT (for Web Focus) ▼

Select Application Directory: baseapp ▼

All Select

Check	Default Synonym Name	Owner/Schema	Table Name	Type
<input type="checkbox"/>	<input type="text" value="NF29001"/>	EDAQA	NF29001	TABLE
<input type="checkbox"/>	<input type="text" value="NF29002"/>	EDAQA	NF29002	TABLE
<input type="checkbox"/>	<input type="text" value="NF29003"/>	EDAQA	NF29003	TABLE
<input type="checkbox"/>	<input type="text" value="NF29004"/>	EDAQA	NF29004	TABLE
<input type="checkbox"/>	<input type="text" value="NF29005"/>	EDAQA	NF29005	TABLE
<input type="checkbox"/>	<input type="text" value="NF29006"/>	EDAQA	NF29006	TABLE
<input type="checkbox"/>	<input type="text" value="NF29007"/>	EDAQA	NF29007	TABLE
<input type="checkbox"/>	<input type="text" value="NF29008"/>	EDAQA	NF29008	TABLE
<input type="checkbox"/>	<input type="text" value="NF29009"/>	EDAQA	NF29009	TABLE
<input type="checkbox"/>	<input type="text" value="NF29012"/>	EDAQA	NF29012	TABLE
<input type="checkbox"/>	<input type="text" value="NF29013"/>	EDAQA	NF29013	TABLE

4. If you have tables with identical table names, assign a prefix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave field blank.

5. From the SET CONVERSION LONGCHAR drop-down list, select: Text, Alpha, or BLOB. The default value is Text. See *Controlling the Mapping of Large Character Data Types* for more information.
6. From the Select Application Directory drop-down list, select a directory. The default value is baseapp.
7. Complete your table or view selection:

To select all tables or views in the list, click **All**.

To select specific tables or views, click the corresponding check boxes.

8. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.
9. Click *Create Synonym*. Synonyms are created and added under the specified application directory.

Reference **Managing Synonyms**

In the navigation pane, click the name of the synonym to access the following options:

Sample Data	Retrieves up to 20 rows from the associated data source.
Properties	Displays a graphic representation of the synonym and enables you to edit its metadata.
Edit Master File	Enables you to manually edit the synonym's Master File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Master File, to update the synonym.
Edit Access File	Enables you to manually edit the synonym's Access File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Access File, to update the synonym.
Refresh	Regenerates the synonym. Use this option if structural changes were made to the data source.
Drop	Deletes the synonym.
Copy to	Copies the synonym to another application directory. Click the target directory from the resulting list.
Move to	Moves the synonym to another application directory. Click the target directory from the resulting list.

Syntax **How to Create a Synonym Manually**

```
CREATE SYNONYM app/synonym FOR datasource DBMS SQLPRO [NOCOLS]
END
```

where:

app

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, then this application name must not be used.

synonym

Is an alias for the data source (maximum 64 characters).

datasource

Is the fully qualified name of the data structure.

SQLPRO

Indicates the Data Adapter for Progress.

AT *connection*

Is the service name as previously specified in a SET CONNECTION_ATTRIBUTES command. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

This parameter is optional. If specified, the CONNECTION attribute is added to the Access File.

NOCOLS

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example Using CREATE SYNONYM

```
CREATE SYNONYM baseapp/nf29004 for EDAQA.NF29004 DBMS SQLPRO
```

Generated Master File nf29004.mas

```
FILE=DIVISION ,SUFFIX=SQLPRO,$
SEGNAME=SEG1_4 ,SEGTYPE=S0 , $
FIELD=DIVISION4 ,DIVISION4,I9 ,I4 , MISSING=OFF,$
FIELD=DIVISION_NA4 ,DIVISION_NA4,A25 ,A25 ,MISSING=ON , $
FIELD=DIVISION_HE4 ,DIVISION_HE4,I9 ,I4 ,MISSING=ON , $
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4 ,TABLENAME=EDAQA.NF29004 ,
KEYS=1, WRITE=YES , $
```

Data Type Support

The following table lists how the server maps Progress data types. Note that you can:

- Control the mapping of large character data types.
- Change the mapping of variable-length data types.
- Change the precision and scale of numeric columns.

Be sure to review those options carefully as they affect the mapping of data types.

Progress Data Type	Data Type		Remarks
	Usage	Actual	
CHAR	A(2000)	A(2000)	
VARCHAR	A(31995)	A(31995)	
DATE	HYYMD	HYYMD	
TIME	HHIS	HHIS	
TIMESTAMP	HYYMDs	HYYMDs	
DECIMAL (p,s)	P14.2	P7	
NUMERIC (p,s)	P14.2	P7	
INTEGER	I11	I4	
SMALLINT	I6	I4	
TINYINT	I6	I4	

Progress Data Type	Data Type		Remarks
	Usage	Actual	
REAL	D20.2	D8	
FLOAT DOUBLE PRECISION	D20.2	D8	
LOGICAL	N/A	N/A	No longer supported under SQL-92.
BIT	I11	I4	
BINARY	A4	A4	
VARBINARY	A20	A20	
LVARBINARY	BLOB	BLOB	

Controlling the Mapping of Large Character Data Types

The SET parameter CONVERSION LONGCHAR controls the mapping of supported Progress data types listed below. By default, the server maps these data types as alphanumeric (A). The server data type A supports a maximum of 4096 characters for TABLE/MODIFY and 32768 characters for API applications.

The following table lists data type mappings based on the value of LONGCHAR:

Progress Data Type	Remarks	LONGCHAR ALPHA or BLOB		LONGCHAR TEXT	
		Usage	Actual	Usage	Actual
CHAR (<i>n</i>)	<i>n</i> is an integer between 1 and 2000	<i>An</i>	<i>An</i>	TX50	TX
VARCHAR (<i>n</i>)	<i>n</i> is an integer between 1 and 4000	<i>An</i>	<i>An</i>	TX50	TX
VARCHAR2 (<i>n</i>)	<i>n</i> is an integer between 1 and 4000	<i>An</i>	<i>An</i>	TX50	TX
RAW (<i>n</i>)	<i>n</i> is an integer between 1 and 2000 $m = 2 * n$	<i>Am</i>	<i>Am</i>	TX50	TX

Syntax **How to Control the Mapping of Large Character Data Types**

`ENGINE [SQLPRO] SET CONVERSION LONGCHAR {ALPHA|TEXT|BLOB}`

where:

SQLPRO

Indicates the Data Adapter for Progress. You can omit this value if you previously issued the SET SQLENGINE command.

ALPHA

Maps the Progress data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as alphanumeric (A). This value is the default.

TEXT

Maps the Progress data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as text (TX). Use this value for WebFOCUS applications.

BLOB

For UNIX, Windows, OpenVMS, and OS/400, is identical to ALPHA. That is, it maps the Progress data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as alphanumeric (A).

For OS/390 and z/OS, maps the Progress data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as binary large object (BLOB).

Controlling the Mapping of Variable-Length Data Types

The SET parameter VARCHAR controls the mapping of the Progress data types VARCHAR, VARCHAR2, and NVARCHAR2. By default, the server maps these data types as alphanumeric (A).

The following table lists data type mappings based on the value of VARCHAR:

Microsoft SQL Server Data Type	Remarks	VARCHAR ON		VARCHAR OFF	
		Usage	Actual	Usage	Actual
VARCHAR (<i>n</i>)	<i>n</i> is an integer between 1 and 4000	<i>AnV</i>	<i>AnV</i>	<i>An</i>	<i>An</i>
VARCHAR2 (<i>n</i>)	<i>n</i> is an integer between 1 and 4000	<i>AnV</i>	<i>AnV</i>	<i>An</i>	<i>An</i>
NVARCHAR2 (<i>n</i>)	<i>n</i> is an integer between 1 and 4000	<i>AmV</i>	<i>AmV</i>	<i>Am</i>	<i>Am</i>

Syntax **How to Control the Mapping of Variable-Length Data Types**

```
ENGINE [SQLPRO] SET VARCHAR {ON|OFF}
```

where:

SQLPRO

Indicates the Data Adapter for Progress. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Maps the Progress data types VARCHAR, VARCHAR2, and NVARCHAR2 as variable-length alphanumeric (AnV).

OFF

Maps the Progress data types VARCHAR, VARCHAR2, and NVARCHAR2 as alphanumeric (A). This value is the default.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Syntax

How to Override Default Precision and Scale

```
ENGINE [SQLPRO] SET CONVERSION RESET  
ENGINE [SQLPRO] SET CONVERSION format RESET  
ENGINE [SQLPRO] SET CONVERSION format [PRECISION pp [ss]]  
ENGINE [SQLPRO] SET CONVERSION format [PRECISION MAX]
```

where:

SQLPRO

Indicates the Data Adapter for Progress. You can omit this value if you previously issued the SET SQLENGINE command.

RESET

Returns any previously specified precision and scale values to the data adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

format

Is any valid format supported by the data source. Possible values are:

INTEGER which indicates that the command applies only to INT columns.

DECIMAL which indicates that the command applies only to DECIMAL columns.

FLOAT which indicates that the command applies only to double precision floating point columns.

pp

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

ss

Is the scale. This is valid with DECIMAL and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. If the scale is not required, you must set *ss* to 0 (zero).

MAX

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE SQLPRO SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLPRO SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE SQLPRO SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLPRO SET CONVERSION RESET
```

Customizing the Progress Environment

The Data Adapter for Progress provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Specifying the Block Size for Array Retrieval

The Data Adapter for Progress supports array retrieval from result sets produced by executing SELECT queries or stored procedures. This technique substantially reduces network traffic and CPU utilization.

Using high values increases the efficiency of requests involving many rows, at the cost of higher virtual storage requirements. A value higher than 100 is not recommended because the increased efficiency it would provide is generally negligible.

Syntax How to Specify the Block Size for Array Retrieval

The block size for a SELECT request applies to TABLE FILE requests, MODIFY requests, MATCH requests, and DIRECT SQL SELECT statements.

```
ENGINE [SQLPRO] SET FETCHSIZE n
```

where:

SQLPRO

Indicates the Data Adapter for Progress. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is the number of rows to be retrieved at once using array retrieval techniques. Accepted values are 1 to 5000. The default is 20. If the result set contains a column that has to be processed as a CLOB or a BLOB, the fetchsize value used for that result set is 1.

Syntax How to Specify the Block Size for Inserting Rows

In combination with LOADONLY, the block size for an INSERT applies to MODIFY INCLUDE requests. INSERTSIZE is also supported for parameterized DIRECT SQL INSERT statements.

```
ENGINE [SQLPRO] SET INSERTSIZE n
```

where:

SQLPRO

Indicates the Data Adapter for Progress. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is the number of rows to be inserted using array insert techniques. Accepted values are 1 to 5000. The default is 1. If the result set contains a column that has to be processed as a BLOB, the insertsize value used for that result set is 1.

Activating NONBLOCK Mode

The Data Adapter for Progress has the ability to issue calls in NONBLOCK mode. The default behavior is BLOCK mode.

This feature allows the adapter to react to a client request to cancel a query while the adapter is waiting on engine processing. This wait state usually occurs during SQL parsing, before the first row of an answer set is ready for delivery to the adapter or while waiting for access to an object that has been locked by another application.

Syntax How to Activate NONBLOCK Mode

```
ENGINE [SQLPRO] SET NONBLOCK {0|n}
```

where:

SQLPRO

Indicates the Data Adapter for Progress. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is a positive numeric number. A value of 0, the default, means that the adapter will operate in BLOCK mode. A value of 1 or greater activates the NONBLOCK calling and specifies the time, in seconds, that the adapter will wait between each time it checks to see if the:

- Query has been executed.
- Client application has requested the cancellation of a query.
- Kill Session button on the Web Console is pressed.

Note: A value of 1 or 2 should be sufficient for normal operations.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Syntax How to Obtain the Number of Rows Updated or Deleted

```
ENGINE [SQLPRO] SET PASSRECS {ON|OFF}
```

where:

SQLPRO

Indicates the Data Adapter for Progress. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. This value is the default.

OFF

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to Progress.

Syntax **How to Issue the TIMEOUT Command**

The syntax is

```
ENGINE [SQLPRO] SET TIMEOUT {nn|0}
```

where:

SQLPRO

Indicates the SQLENGINE setting. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a timeout occurs. The default value is 30.

0

Represents an infinite period to wait for response.

CHAPTER 30

Getting Started in Rdb

Topics:

- Preparing the Rdb Environment
- Configuring the Data Adapter for Rdb
- Managing Rdb Metadata
- Joining Rdb Tables in Separate Rdb Databases
- Rdb Database Driver Performance

The Data Adapter for Rdb allows applications to access Rdb data sources. The adapter converts application requests into native Rdb statements and returns optimized answer sets to the requesting application.

Preparing the Rdb Environment

Oracle Rdb allows two types of installation:

- **Standard.** Supports only one Rdb release level on a machine. No preparation is required to use the adapter in a Standard Rdb environment.
- **Multi Version.** Supports multiple Rdb release levels on the same machine. To switch between releases, Multi Version supplies a script, which sets up logically to point standard names, such as SQL\$, at specific release files, such as SQL\$61.EXE, SQL\$70.EXE, and SQL\$7.1.EXE.

To use the adapter in a Multi Version environment, prior to server startup, your Rdb environment must be set up to access the database file release level that corresponds to the release level selected during configuration of the adapter. To enable switching between Rdb releases, your site might use the Rdb script DECRDB\$SETVER, a site-specific script, or a symbol. For details on how Rdb release level switching is implemented at your site, see your OpenVMS Administrator.

Configuring the Data Adapter for Rdb

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

In order to connect to an Rdb database server, the adapter requires connection and authentication information. You supply this information using the SET SERVER command. You can:

- Manually add the command in the global server profile (edasprof.prf) or in a user profile (user.prf).
- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the commands to the global server profile.

Syntax **How to Declare Connection Attributes Manually**

The adapter connects to Rdb using the credentials of the operating system user impersonated by the data access agent. The syntax is

```
ENGINE [SQLRDB] SET SERVER server
```

where:

SQLRDB

Indicates the Data Adapter for Rdb. You can omit this value if you previously issued the SET SQLENGINE command.

server

Is the full path to an Rdb data source, or a logical name, such as SQL\$DATABASE, which contains the full path. Rdb limits this value to 31 characters. If the full path exceeds this limit, you must specify a logical name to point to the full path.

Full Path. Full path specification must be in the following form: *device_name:[directory]file.RDB*. The RDB extension is required.

Logical Name. To use this method, the logical must be set prior to server startup. The trailing blank is optional.

Note: You can manually declare connections to more than one RDB database server by including multiple SET SERVER commands. The actual connection to RDB takes place when the first query that references the connection is issued. If you issue multiple SET SERVER commands:

- The connection named in the last SET SERVER command serves as the default connection.
- The RDB installation has to be Standard.

Example **Declaring a Connection to ORDERS.RDB**

The following example shows how to declare a connection to the ORDERS.RDB data source, located in the OUTBOUND directory on disk.

```
ENGINE SQLRDB SET SERVER DISK$SHIPPING:[OUTBOUND]ORDERS.RDB
```

Reference Declaring Rdb Connection Attributes From the Web Console

Attribute	Description
Server	<p>Is the full path to an Rdb data source, or a logical name, such as SQL\$DATABASE, which contains the full path. Rdb limits this value to 31 characters. If the full path exceeds this limit, you must specify a logical name to point to the full path.</p> <p>Full Path. Full path specification must be in the following form: <i>device_name:[directory]file.RDB</i>. The RDB extension is required.</p> <p>Logical Name. To use this method, the logical must be set prior to server startup. The trailing blank is optional.</p>

Overriding the Default Connection

Once connections have been defined, the connection named in the last SET SERVER command serves as the default connection. You can override this default using the command.

Syntax How to Change the Default Connection

```
ENGINE [SQLRDB] SET SERVER [server]
```

where:

SQLRDB

Indicates the Data Adapter for Rdb. You can omit this value if you previously issued the SET SQLENGINE command.

server

Is the full path to an Rdb data source, or a logical name, such as SQL\$DATABASE, which contains the full path. If this name was not previously declared, the following error message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET SERVER command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET SERVER command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following error message is issued: FOC1671, Command out of sequence.

Example **Selecting the Default Connection**

The following SET SERVER command selects the Rdb database server named SAMPLENAME as the default Rdb database server:

```
ENGINE SQLRDB SET SERVER SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax **How to Control the Connection Scope**

```
ENGINE [SQLRDB] SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

SQLRDB

Indicates the Data Adapter for Rdb. You can omit this value if you previously issued the SET SQLENGINE command.

FIN

Disconnects automatically only after the session has been terminated. This value is the default.

COMMIT

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Rdb Metadata

When the server accesses a data source, it needs to know how to interpret the data that it finds. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Rdb data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Rdb table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click Metadata. The Editing Metadata page opens.

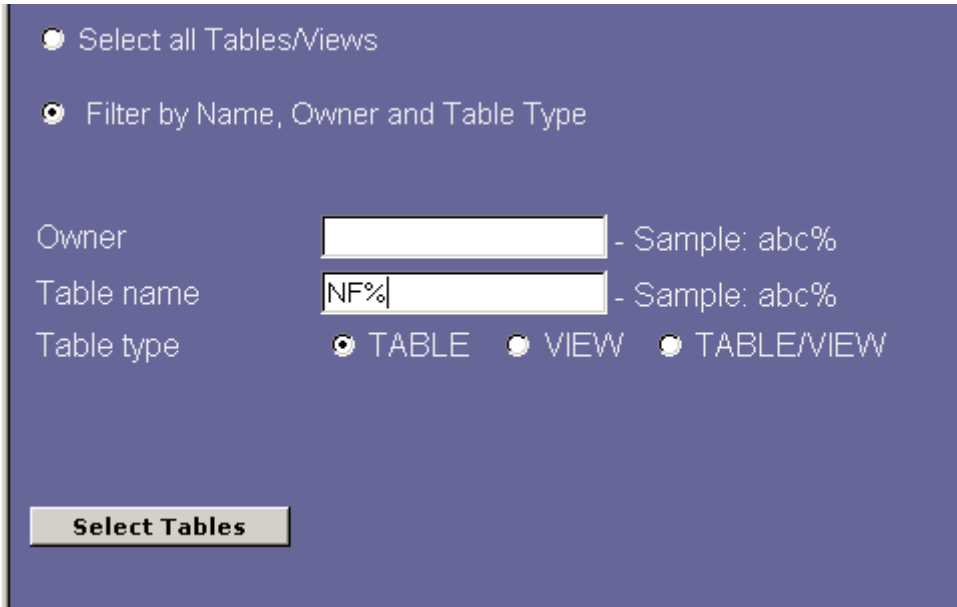
To create a synonym, you must have configured the adapter. See *Configuring the Data Adapter for Rdb* for more information.

2. Expand the *Add* folder, expand the adapter folder, and then click a connection. The right pane displays table and view selection options:

Select All Tables/Views. Select this radio button to create synonyms for all tables and views. This value is the default.

Filter by Name, Owner and Table Type. Select this radio button to filter the tables or views for which to create synonyms.

Selecting this option adds the following:



Select all Tables/Views

Filter by Name, Owner and Table Type

Owner - Sample: abc%

Table name - Sample: abc%

Table type TABLE VIEW TABLE/VIEW

Select Tables

Owner. Enter a string for filtering the owners' IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owners' IDs begin with the letters ABC; %ABC to select tables or views whose owners' IDs end with the letters ABC; %ABC% to select tables or views whose owners' IDs contain the letters ABC at the beginning, middle, or end.

Table Name. Enter a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

Table Type. Select one of the following options: Tables (this is the default), Views, Tables/Views.

3. Click *Select Tables*. All tables that meet the specified criteria are displayed:

Prefix: SET CONVERSION LONGCHAR:

Select Application Directory:

All Select

Check	Default Synonym Name	Owner/Schema	Table Name	Type
<input type="checkbox"/>	<input type="text" value="NF29001"/>	EDAQA	NF29001	TABLE
<input type="checkbox"/>	<input type="text" value="NF29002"/>	EDAQA	NF29002	TABLE
<input type="checkbox"/>	<input type="text" value="NF29003"/>	EDAQA	NF29003	TABLE
<input type="checkbox"/>	<input type="text" value="NF29004"/>	EDAQA	NF29004	TABLE
<input type="checkbox"/>	<input type="text" value="NF29005"/>	EDAQA	NF29005	TABLE
<input type="checkbox"/>	<input type="text" value="NF29006"/>	EDAQA	NF29006	TABLE
<input type="checkbox"/>	<input type="text" value="NF29007"/>	EDAQA	NF29007	TABLE
<input type="checkbox"/>	<input type="text" value="NF29008"/>	EDAQA	NF29008	TABLE
<input type="checkbox"/>	<input type="text" value="NF29009"/>	EDAQA	NF29009	TABLE
<input type="checkbox"/>	<input type="text" value="NF29012"/>	EDAQA	NF29012	TABLE
<input type="checkbox"/>	<input type="text" value="NF29013"/>	EDAQA	NF29013	TABLE

4. If you have tables with identical table names, assign a prefix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave field blank.

5. From the Select Application Directory drop-down list, select a directory. The default value is baseapp.

6. Complete your table or view selection:
 To select all tables or views in the list, click *All*.
 To select specific tables or views, click the corresponding check boxes.
7. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.
8. Click *Create Synonym*. Synonyms are created and added under the specified application directory.

Reference **Managing Synonyms**

In the navigation pane, click the name of the synonym to access the following options:

Sample Data	Retrieves up to 20 rows from the associated data source.
Properties	Displays a graphic representation of the synonym and enables you to edit its metadata.
Edit Master File	Enables you to manually edit the synonym's Master File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Master File, to update the synonym.
Edit Access File	Enables you to manually edit the synonym's Access File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Access File, to update the synonym.
Refresh	Regenerates the synonym. Use this option if structural changes were made to the data source.
Drop	Deletes the synonym.
Copy to	Copies the synonym to another application directory. Click the target directory from the resulting list.
Move to	Moves the synonym to another application directory. Click the target directory from the resulting list.

Syntax **How to Create a Synonym Manually**

```
CREATE SYNONYM app/synonym FOR datasource DBMS SQLRDB [AT server][NOCOLS]
                                     [AT '']
END
```

where:

app

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, then this application name must not be used.

synonym

Is an alias for the data source (maximum 64 characters).

datasource

Is the Rdb data source as specified in the SET SERVER command. If a logical name was used in SET SERVER, the same logical name must be provided for CREATE SYNONYM also.

SQLRDB

Indicates the Data Adapter for Rdb.

AT *server*

Is the service name as previously specified in a SET SERVER command. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

This parameter is optional. If specified, the CONNECTION attribute is added to the Access File.

AT ' '

Adds CONNECTION=' ' in the Access File. This indicates a connection to a local Rdb database server previously specified in the SET SERVER declaration.

NOCOLS

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.NF29004 DBMS SQLRDB AT DSN_A
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION ,SUFFIX=SQLRDB,$
SEGNAME=SEG1_4 ,SEGTYPE=S0 ,$
FIELD=DIVISION4 ,DIVISION4 ,I9 ,I4 ,MISSING=OFF ,$
FIELD=DIVISION_NA4,DIVISION_NA4 ,A25 ,A25 ,MISSING=ON ,$
FIELD=DIVISION_HE4,DIVISION_HE4 ,I9 ,I4 ,MISSING=ON ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4, TABLENAME=EDAQA.NF29004,
CONNECTION=DSN_A, KEYS=1, WRITE=YES, $
```

Reference Access File Keywords

Keyword	Description
TABLENAME	Identifies the Rdb table.
CONNECTION	Indicates a previously declared connection. The syntax is: <i>CONNECTION=connection</i> CONNECTION='' indicates access to the local Rdb database server. Absence of the CONNECTION attribute indicates access to the default database server.

Data Type Support

The following chart provides information about the default mapping of Rdb data types to server data types:

Rdb Data Type	Data Type		Remarks
	Usage	Actual	
CHAR (n)	An	An	
VARCHAR (n)	An	An	
LONGVARCHAR	An	An	
TINYINT (-128...127)	I6	I4	Scale factors (n) in SQL integer data types are equivalent to negative scale factors in Oracle Rdb integer data types. SQL does not support Oracle Rdb positive scale factors.
SMALLINT (-32768... 32767)	I6	I4	
INTEGER (-2 ³¹ ... 2 ³¹ - 1)	I11	I4	2 ³¹ = 2,147,483,648.
NUMERIC, DECIMAL (p,s)	D20.2	D8	Same as REAL or DOUBLE PRECISION. 8 bytes; value range 2 * 10 ⁺³⁰⁷ to 2 * 10 ⁺³⁰⁸ .
BIGINT (-2 ⁶³ ... 2 ⁶³ - 1)	D20.2	D8	
REAL	D20.2	D8	
DOUBLE PRECISION	D20.2	D8	
DATE	An	An	Date/Time formats are comprised of several components.
TIME	An	An	Date/Time formats are comprised of several components.
TIMESTAMP	An	An	Date/Time formats are comprised of several components.

Joining Rdb Tables in Separate Rdb Databases

In iWay 5.x for OpenVMS, two or more Rdb tables that exist in different physical Rdb databases may be joined. A default iWay configuration can access only one physical Rdb database at a time. The use of multiple databases is accomplished by doing a standard configuration with Rdb for a given database, creating metadata for that database, repeating the configure/metadata step for as many Rdb databases as needed, and making changes to a server's configuration as outlined below:

1. During server configuration, choose the standard option for Rdb relational access and supply the full path name to one of the desired Rdb databases.
2. Edit the EDACONF [.BIN]EDAENV.COM file and add a unique logical name for each desired database file to be accessed. Specify the full path names to each of the databases. For example:

```
$ DEFINE RDBSRV1 DISK$DUA0:[RDB]SCORES.RDB
```

```
$ DEFINE RDBSRV2 DISK$DUA0:[RDB]STUDENTS.RDB
```

3. Assuming that there are previously created Rdb Access files, edit the Access Files for each of the Rdb databases, and change the TABLENAME value to be that of the appropriate logical name. Use a period and the original value (for example, logical.TNAME), and add a parameter of SERVER=logical.

The following is an example of the respective Access File contents before changes:

```
..., TABLENAME=SCORES, ...
```

```
..., TABLENAME=STUDENTS, ...
```

An example of the respective Access File contents after changes is

```
..., TABLENAME=RDBSRV1.SCORES, SERVER=RDBSRV1, ...
```

```
..., TABLENAME=RDBSRV2.STUDENTS, SERVER=RDBSRV2, ...
```

where:

RDBSRV1

Is a unique logical for the database in question.

RDBSRV2

Is a unique logical for the database in question.

SCORES

Is a table name.

STUDENTS

Is a table name.

4. Edit the EDASPROF.PRF file, comment out the SET SERVER command for the database specified during configuration, and add:

```
SQL
SET APT=OFF
END
```

FOCUS JOIN statements or SQL JOIN syntax can now be used to retrieve data. This access method is not supported for use with direct passthru (for example, SET SQLENGINE=SQLRDB) to join the tables. If you wish to use direct passthru for access to one Rdb database, you will need to add back the SET SERVER command in EDASPROF.PRF for that database.

Rdb Database Driver Performance

The default of Rdb is to open tables in read/write mode unless told otherwise. iWay supports read/write operations, so this is appropriate behavior. However, Rdb read/write opens consume more resources and are slower even if just a select is being done. iWay applications can set Rdb opens to Read-only on a per request basis or a per session basis to enhance resource usage and speed performance. In either case, a commit should be done to ensure that prior work is complete.

On a session basis, add the following to the [EDASPROF.PRF](#) after the SQL SQLRDB SET SERVER command:

```
SQLRDB COMMIT ;
SQL SQLRDB SET DECLARE TRANSACTION READ ONLY ;
```

On a per request basis (for example, immediately before a TABLE request) the usage is:

```
SQLRDB COMMIT ;
SQL SQLRDB SET TRANSACTION READ ONLY ;
```

CHAPTER 31

Getting Started in Red Brick

Topics:

- Preparing the Red Brick Environment
- Configuring the Data Adapter for Red Brick
- Managing Red Brick Metadata
- Customizing the Red Brick Environment

The Data Adapter for Red Brick allows applications to access Red Brick data sources. The adapter converts application requests into native Red Brick statements and returns optimized answer sets to the requesting application.

Preparing the Red Brick Environment

The Red Brick Data Adapter minimally requires the installation of the IBM Red Brick Client 32 ODBC software. After installation, ODBC must be configured to allow you to connect to a local or remote Red Brick database server.

Procedure How to Set Up the Environment on Windows NT/2000

On Windows NT/2000, the Red Brick environment is set up during the installation of Red Brick.

Procedure How to Set Up the Environment on UNIX

1. Specify the location of the Red Brick database you wish to access using the UNIX environment variable `$RB_CONFIG`. For example, to set the home directory for the Red Brick software to `/rdbms/red61`, specify:

```
RB_CONFIG=/rdbms/red61
export RB_CONFIG
```

2. Define the `$HOME/.odbc.ini` file, which describes the Red Brick data sources. You can use the `ODBCINI` variable instead of `$HOME/.odbc.ini`.

Configuring the Data Adapter for Red Brick

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to an Red Brick database server, the adapter requires connection and authentication information. You supply this information using the `SET CONNECTION_ATTRIBUTES` command. You can:

- Manually add the command in the global server profile (`edasprof.prf`) or in a user profile (`user.prf`).
- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the global server profile. For more information, see *Declaring Red Brick Connection Attributes From the Web Console* on page 31-4.

You can declare connections to more than one Red Brick database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to Red Brick Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Syntax **How to Declare Connection Attributes Manually**

```
ENGINE [SQLRED] SET CONNECTION_ATTRIBUTES [connection]/user_ID,password
```

where:

SQLRED

Indicates the Data Adapter for Red Brick. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the name of the Red Brick DSN (Data Source Name) you wish to access. It must match an entry in the .odbc.ini.

user_ID

Is the primary authorization ID by which you are known to Red Brick.

password

Is the password associated with the primary authorization ID.

Example **Declaring Connection Attributes**

The following SET CONNECTION_ATTRIBUTES command connects to the Red Brick database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLRED SET CONNECTION_ATTRIBUTES SAMPLESERVER/MYUSER,PASS
```

Reference **Declaring Red Brick Connection Attributes From the Web Console**

Attribute	Description
Data source	Is the Red Brick data source name (DSN). There is no default data source name. You must enter a value.
User	Is the authorization by which the user is known to Red Brick
Password	Is the password associated with the authorization ID. The password is stored in encrypted form.

Overriding the Default Connection

Once connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax How to Change the Default Connection

```
ENGINE [SQLRED] SET DEFAULT_CONNECTION [connection]
```

where:

SQLRED

Indicates the Data Adapter for Red Brick. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the service name defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following error message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following error message is issued: FOC1671, Command out of sequence.

Example Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the Red Brick database server named SAMPLENAME as the default Red Brick database server:

```
ENGINE SQLRED SET DEFAULT_CONNECTION SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax **How to Control the Connection Scope**

```
ENGINE [SQLRED] SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

SQLRED

Indicates the Data Adapter for Red Brick. You can omit this value if you previously issued the SET SQLENGINE command.

FIN

Disconnects automatically only after the session has been terminated. This value is the default.

COMMAND

Disconnects automatically after each request. Depending on how often the event occurs, the AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server; it is related to the operating system and the data source.

COMMIT

Disconnects automatically only after COMMIT is issued as a native SQL command.

Managing Red Brick Metadata

When the server accesses a data source, it needs to know how to interpret the data that it finds. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Red Brick data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Red Brick table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Editing Metadata page opens.

To create a synonym, you must have configured the adapter. See *Configuring the Data Adapter for Red Brick* on page 31-2 for more information.

2. Expand the *Add* folder, expand the adapter folder, and then click a connection. The right pane displays table and view selection options:

Select All Tables/Views. Select this radio button to create synonyms for all tables and views. This value is the default.

Filter by Name, Owner and Table Type. Select this radio button to filter the tables or views for which to create synonyms.

Selecting this option adds the following:

The screenshot shows a web console interface with a dark blue background. At the top, there are two radio button options: 'Select all Tables/Views' (unselected) and 'Filter by Name, Owner and Table Type' (selected). Below these are three input fields: 'Owner' (empty), 'Table name' (containing 'NF%'), and 'Table type' (with three radio buttons: 'TABLE' selected, 'VIEW' unselected, and 'TABLE/VIEW' unselected). At the bottom left, there is a button labeled 'Select Tables'.

Owner. Enter a string for filtering the owners' IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owners' IDs begin with the letters ABC; %ABC to select tables or views whose owners' IDs end with the letters ABC; %ABC% to select tables or views whose owners' IDs contain the letters ABC at the beginning, middle, or end.

Table Name. Enter a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

Table Type. Select one of the following options: Tables (this is the default), Views, Tables/Views.

3. Click *Select Tables*. All tables that meet the specified criteria are displayed:

Prefix: SET CONVERSION LONGCHAR:

Select Application Directory:

All Select

Check	Default Synonym Name	Owner/Schema	Table Name	Type
<input type="checkbox"/>	<input type="text" value="NF29001"/>	EDAQA	NF29001	TABLE
<input type="checkbox"/>	<input type="text" value="NF29002"/>	EDAQA	NF29002	TABLE
<input type="checkbox"/>	<input type="text" value="NF29003"/>	EDAQA	NF29003	TABLE
<input type="checkbox"/>	<input type="text" value="NF29004"/>	EDAQA	NF29004	TABLE
<input type="checkbox"/>	<input type="text" value="NF29005"/>	EDAQA	NF29005	TABLE
<input type="checkbox"/>	<input type="text" value="NF29006"/>	EDAQA	NF29006	TABLE
<input type="checkbox"/>	<input type="text" value="NF29007"/>	EDAQA	NF29007	TABLE
<input type="checkbox"/>	<input type="text" value="NF29008"/>	EDAQA	NF29008	TABLE
<input type="checkbox"/>	<input type="text" value="NF29009"/>	EDAQA	NF29009	TABLE
<input type="checkbox"/>	<input type="text" value="NF29012"/>	EDAQA	NF29012	TABLE
<input type="checkbox"/>	<input type="text" value="NF29013"/>	EDAQA	NF29013	TABLE

4. If you have tables with identical table names, assign a prefix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave field blank.

5. From the SET CONVERSION LONGCHAR drop-down list, select: Text, Alpha, or BLOB.

6. From the Select Application Directory drop-down list, select a directory. The default value is baseapp.
7. Complete your table or view selection:
To select all tables or views in the list, click **All**.
To select specific tables or views, click the corresponding check boxes.
8. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.
9. Click *Create Synonym*. Synonyms are created and added under the specified application directory.

Reference **Managing Synonyms**

In the navigation pane, click the name of the synonym to access the following options:

Sample Data	Retrieves up to 20 rows from the associated data source.
Properties	Displays a graphic representation of the synonym and enables you to edit its metadata.
Edit Master File	Enables you to manually edit the synonym's Master File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Master File, to update the synonym.
Edit Access File	Enables you to manually edit the synonym's Access File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Access File, to update the synonym.
Refresh	Regenerates the synonym. Use this option if structural changes were made to the data source.
Drop	Deletes the synonym.
Copy to	Copies the synonym to another application directory. Click the target directory from the resulting list.
Move to	Moves the synonym to another application directory. Click the target directory from the resulting list.

Syntax **How to Create a Synonym Manually**

```
CREATE SYNONYM app/synonym FOR table_view DBMS SQLRED [AT connection]  
[NOCOLS]  
END
```

where:

app

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, then this application name must not be used.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema) as follows:

```
[owner.] table
```

SQLRED

Indicates the Data Adapter for Red Brick.

AT *connection*

Is the *connection_name*, as previously specified in a SET CONNECTION_ATTRIBUTES command. When the server creates the synonym, this *connection_name* becomes the value for CONNECTION= in the Access File.

For Red Brick, this is the Data Source Name (DSN) as it is defined in \$HOME/.odbc.ini.

If, after creating the synonyms with this option, the DBMS server's environment changes (for example, if tables are moved to a different DBMS server), you must either create new synonyms or edit existing ones.

If this parameter is omitted, the server uses DBMS specific settings in its environment.

NOCOLS

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.NF29004 DBMS SQLRED AT DSN_A
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION ,SUFFIX=SQLRED,$
SEGNAME=SEG1_4 ,SEGTYPE=S0 ,$
FIELD=DIVISION4 ,DIVISION4 ,I9 ,I4 ,MISSING=OFF ,$
FIELD=DIVISION_NA4,DIVISION_NA4 ,A25 ,A25 ,MISSING=ON ,$
FIELD=DIVISION_HE4,DIVISION_HE4 ,I9 ,I4 ,MISSING=ON ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4 ,TABLENAME=EDAQA.NF29004 ,
CONNECTION=DSN_A ,KEYS=1 ,WRITE=YES , $
```

Reference Access File Keywords

Keyword	Description
TABLENAME	Identifies the Red Brick table. The value assigned to this attribute can include the name of the owner (also known as schema) as follows: <code>TABLENAME=[owner.]table</code>
CONNECTION	Indicates a previously declared connection. The syntax is: <code>CONNECTION=connection</code> CONNECTION=' ' indicates access to the local Red Brick database server. Absence of the CONNECTION attribute indicates access to the default database server.

Data Type Support

The following table lists how the server maps Red Brick data types.

Red Brick Data Type	Data Type		Remarks
	Usage	Actual	
CHAR (<i>n</i>)	<i>An</i>	<i>An</i>	<i>n</i> is an integer between 1 and 2000
VARCHAR (<i>n</i>)	<i>An</i>	<i>An</i>	<i>n</i> is an integer between 1 and 4000
DATE	HYYMD	HYYMD	Date/Time formats are comprised of several components. Not a true data value stored internally.
TIMESTAMP	HYYMDm	HYYMDm	Date/Time formats are comprised of several components. Not a true data value stored internally.
TINYINT	I6	I4	
SMALLINT	I6	I4	
REAL	D20.2	D8	
DOUBLE FLOAT	D20.2	D8	
SERIAL	P20.2	P8 P10	The SERIAL data type is a special case of the INTEGER data type. There can be only one SERIAL column in any table. SERIAL data types are not allowed in pre-computed view definitions, including view columns, view table columns, and predicates. When defined, MUST BE specified as NOT NULL.
INTEGER	I11	I4	$2^{31} = 2,147,483,648$
NUMERIC, DECIMAL	D20.2	D8	Same as REAL or DOUBLE PRECISION

Controlling the Mapping of Variable-Length Data Types

The SET parameter VARCHAR controls the mapping of the Red Brick data type VARCHAR. By default, the server maps this data type as alphanumeric (A).

Syntax How to Control the Mapping of Variable-Length Data Types

```
ENGINE [SQLRED] SET VARCHAR {ON|OFF}
```

where:

SQLRED

Indicates the Data Adapter for Red Brick. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Maps the Red Brick data type VARCHAR as variable-length alphanumeric (AnV).

OFF

Maps the Red Brick data type VARCHAR as alphanumeric (A). This value is the default.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Syntax How to Override Default Precision and Scale

```
ENGINE [SQLRED] SET CONVERSION RESET
ENGINE [SQLRED] SET CONVERSION format RESET
ENGINE [SQLRED] SET CONVERSION format [PRECISION pp [ss]]
ENGINE [SQLRED] SET CONVERSION format [PRECISION MAX]
```

where:

SQLRED

Indicates the Data Adapter for Red Brick. You can omit this value if you previously issued the SET SQLENGINE command.

RESET

Returns any previously specified precision and scale values to the data adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

format

Is any valid format supported by the data source. Possible values are:

INTEGER which indicates that the command applies only to INT columns.

DECIMAL which indicates that the command applies only to DECIMAL columns.

FLOAT which indicates that the command applies only to double precision floating point columns.

pp

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

ss

Is the scale. This is valid with DECIMAL and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. If the scale is not required, you must set mm to 0 (zero).

MAX

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE SQLRED SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLRED SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE SQLRED SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLRED SET CONVERSION RESET
```

Customizing the Red Brick Environment

The Data Adapter for Red Brick provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to Red Brick.

Syntax How to Issue the TIMEOUT Command

The syntax is

```
ENGINE [SQLRED] SET TIMEOUT {nn|0}
```

where:

SQLRED

Indicates the SQLENGINE setting. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a timeout occurs. The default value is 30.

0

Represents an infinite period to wait for response.

CHAPTER 32

Getting Started in RMS

Topics:

- Manually Describing RMS Files
- Describing Complex RMS Keyed
- Assigning a Keyed RMS File to a Master File
- Retrieving Data From RMS Files
- Syntax for RMS Master File Attributes
- RMS Attribute Summary
- Read/Write Usage Limitations of the Data Adapter for RMS

The Data Adapter for RMS allows applications to access RMS data sources. The adapter converts application requests into native RMS statements and returns answer sets to the requesting application. If the metadata is configured for read/write capabilities, it can insert data from an application to the data source.

These topics discuss how to create metadata for access to RMS files.

You describe data files to the server for OpenVMS with a Master and an Access File. These are comma-delimited files you create with a system editor or with the RMS Automatic Description Generation facility. A Master File describes the type of data file you are using, the structure, and the fields it contains. An Access File associates the Master File to the RMS file.

Note: For information on installing and using the RMS Automatic Description Generation facility, see the *iWay OpenVMS I52Tools* documentation.

Manually Describing RMS Files

These topics outline the procedures necessary for manually describing RMS files.

File Attributes

File attributes are FILENAME and SUFFIX, which name the file and describe the file type. For example:

```
FILENAME=LIBRARY1 , SUFFIX=RMS,$
```

FILENAME

The FILENAME attribute is optional. It is recommended that you include the file name for documentation purposes. You can specify any name for this attribute, but you should use the same name that you give the Master File.

The syntax is

```
FILE[NAME] = name
```

where:

name

Is any name of 1 to 8 characters.

SUFFIX

The SUFFIX attribute describes the type of data file it will read.

The syntax is:

```
[FILE]SUFFIX = type
```

where:

type

Is a suffix listed in the following table:

Type of File	Suffix	Physical Access Method
Keyed (Indexed) RMS file	RMS	Access File
Fixed-format sequential RMS file	FIX	FILEDEF
Comma-delimited sequential RMS file	COM	FILEDEF

The description of data and relationships between fields within a file is the same for keyed (indexed) RMS files, FIX (fixed-format sequential) files, and COM (comma-delimited) files, except for the following differences:

- Keyed (indexed) RMS uses the keyed RMS File System to access data using information in the Access File declarations. You can override the Access File declaration with a FILEDEF. For more information, see *Assigning a Keyed RMS File to a Master File* on page 32-33. The others use sequential access using a FILEDEF to identify and access the file. The coding of FILEDEFs may be done locally within a procedure (before the access request), or within the EDASPROF.PRF if commonly accessed by multiple users.
- Index related declarations do not apply to fixed-sequential or comma-delimited files.

Segment Attributes

A Master File can be divided into segments, which are groups of fields that are related to one another. It is not always necessary to divide your file into segments.

Segment declarations identify and describe each segment in a file. They name the segments and indicate the relative positions in the file structure. In files with multiple record types, each record type will be described as a separate segment. Files with mixed singly- and multiply-occurring fields must also be described with separate segments defined for each type of occurrence. The attributes used to describe segments of different record types and multiply-occurring fields are described in *Describing Multiple Record Types* on page 32-17 and in *Describing Embedded Repeating Data* on page 32-23.

The following is an example of a segment declaration:

```
SEGNAME=BOOKINFO, PARENT=PUBINFO, SEGTYPE=S0,$
```

SEGNAME

Each segment declaration starts with the SEGNAME (or SEGMENT) attribute, which names the segment.

The syntax is

```
{SEGNAME | SEGMENT} = name
```

where:

name

Is a unique name of 1 to 8 characters.

PARENT

Files with more than one segment are defined as multi-segment structures.

Segments in a multi-segment structure have a parent/child relationship. Each segment, except the top or “root” segment, is the descendant of another segment called the “parent.” The PARENT attribute is used to identify a segment's parent segment. If no PARENT attribute is specified in the Master File, the default parent segment is the immediately preceding segment, except for the top segment, which has no parent.

The syntax is

```
PARENT = name
```

where:

name

Specifies a SEGNAME in the file.

For example:

```
...PARENT=PUBINFO...
```

SEGTYPE

The SEGTYPE attribute for RMS files is specified as S0.

The required syntax is:

```
SEGTYPE=S0
```

Field Attributes

Field attributes describe the actual fields in each segment. Each field declaration consists of at least four attributes. They are:

```
FIELDNAME ALIAS USAGE ACTUAL
```

For example:

```
FIELDNAME=PUBNO ,ALIAS=PN ,USAGE=A10 ,ACTUAL=A10 , $
```

There are also other optional attributes that can be used, such as DESCRIPTION, TITLE, and ACCEPT. DESCRIPTION enables you to include a description of the field. TITLE is the default report column title other than the field name. ACCEPT assigns a list or range of acceptable values to a field. ACCEPT is also used for RECTYPE values. These optional attributes are used by FOCUS, WebFOCUS, and various other client products. For further information about optional attributes, see the appropriate manuals.

FIELDNAME

Field names are unique names of 1 to 66 characters with the exception of indexes, which are limited to 12. The field name appears as a default column heading when you name the field in a report request. Field names may consist of any alphanumeric characters, but the first character must be a letter from A to Z. Field names may include embedded blanks, but it is not recommended, and you will need to enclose such a field name in single quotation marks (') if you reference the complete name (including the blank) in a request. Avoid using special characters (+ - \$ * /() ' ; . , = and " > <), since they may cause confusion if the field is used in calculations.

The syntax is

```
FIELD[NAME] = name
```

where:

name

Meets the criteria described above.

ALIAS

Aliases are optional field names. Each field can have an alias to be used interchangeably with the field name. The length and format rules for field names apply to aliases. Aliases are not used as column titles.

The syntax is

```
ALIAS = alias
```

where:

alias

Is a name of 1 to 66 alphanumeric characters meeting the same criteria as for field names.

If you omit the alias, you must indicate its absence with either the following entry in the field declaration

```
ALIAS=,
```

or by holding its place with a comma delimiter (,):

```
FIELDNAME=PUBNO , ,USAGE=A10 ,ACTUAL=A10 ,,$
```

USAGE

The USAGE attribute describes the way you want to use the field and display its values on reports. This attribute includes the data field type, display length, and any edit options that are to be applied when the field values are printed.

The syntax is

`USAGE = format`

where:

format

Describes the field in three parts: field type, field display length, and edit options.

The values that you specify for type and display length determine the number of print positions allocated for the field in any display or report. Edit options only affect printed or displayed fields; they are not active for extract files or other non-display retrievals.

The following table shows permissible USAGE field types and display lengths:

USAGE	Length	Description
A	1-9,095	Alphanumeric text
D	1-19	Decimal, double-precision numbers
F	1-9	Decimal, single-precision numbers
I	1-11	Integer values (no decimal places)
P	1-17	Packed decimal numbers
YYMD	10	Displayed as YYMD
D, W, M, Q or Y	Date	Date display

The following table shows edit options that may be applied to various A, D, F, I, or P usages:

Edit Option	Meaning	Effect
%	Percent sign	Percent sign Displays a percent sign along with numeric data. Does not calculate the percent.
B	Bracket negative	Encloses negative numbers in parentheses.
c	Comma suppress	Suppresses the display of commas. Used with numeric format options M and N (floating and non-floating dollar sign) and data format D (floating-point double-precision).
C	Comma edit	Inserts a comma after every third significant digit, or a period instead of a comma if continental decimal notation is in use.
DMY	Day-Month-Year	Displays alphanumeric or integer data as a date in the form day/month/year.
E	Scientific notation	Scientific notation Displays only significant digits.
L	Leading zeroes	Adds leading zeroes.
M	Floating \$ (for US code page)	Places a floating dollar sign \$ to the left of the highest significant digit. Note: The currency symbol displayed depends on the code page used.
MDY	Month-Day-Year	Displays alphanumeric or integer data as a date in the form month/day/year.
N	Fixed \$ (for US code page)	Places a dollar sign \$ to the left of the field. The symbol displays only on the first detail line of each page. Note: The currency symbol displayed depends on the code page used.
R	Credit (CR) negative	Places CR after negative numbers.
S	Zero suppress	Zero suppress If the data value is zero, prints a blank in its place.

Note:

- Edit options can be specified in any order.
- The total length of the USAGE specification, including all edit options, may not exceed eight characters.
- Options M and N (floating and fixed dollar sign) both automatically imply option C (comma edit).
- Format type D (double-precision decimal number) implies option C.

The following table shows the various USAGE formats as they would be specified in a Master File. The USAGE formats contain type and length information as well as various edit options. The "Value" column shows the actual value as it would be read from the external file, and the "Display" column shows how the number would be displayed on a report.

USAGE	Value	Display
I7C	47693	47,693
D10.2S	0.00	
P8.0CR	-4719	4,719 CR
F8.2MC	28148.00	\$28,148.00
I5B	-341	(341)
D7M	8741	\$8,741
D7N	8741	\$8,741
P6L	21	000021
D12.5	E1234.56	0.123456D+04
I6MDY	20675	02/06/75
A6YMD	750601	75/06/01
A10	HELLO	HELLO
A10	HELLO	HELLO

ACTUAL

The ACTUAL attribute describes the type and length of your data as it actually exists in a file. The source of this information is the existing description of the file.

The syntax is

ACTUAL = format

where:

format

Is any of the format types listed in the following table:

ACTUAL Type	Definition
<i>An</i>	Alphanumeric characters A to Z, 0 to 9, and other displayable ASCII characters, where <i>n</i> equals 1 to 4,095.
<i>D8</i>	Double-precision floating-point numbers stored internally in eight bytes (D_Floating).
<i>F4</i>	Single-precision floating-point numbers stored internally in four bytes (F_Floating).
<i>In</i>	<p>Binary integers:</p> <p>I1 Single-byte signed, binary integer (signed byte)</p> <p>I2 2-byte signed, binary integer (signed word)</p> <p>I4 4-byte signed, binary integer (signed longword)</p> <p>I8 8-byte signed, binary integer (signed quadword)</p> <p>If an integer field contains an assumed decimal point (that is, if it is a scaled integer), represent the field as <i>Im.n</i>, where <i>m</i> is the total number of storage bytes, and <i>n</i> is the number of decimal places. Therefore, <i>I4.1</i> means a 4-byte number with one decimal place.</p>
<i>Pn</i>	Packed decimal format (packed numeric string) where <i>n</i> is the number of bytes (1 to 16), each of which contains two digits, except for the last byte, which contains a digit and the sign. For example, P6 means 11 digits plus a sign, packed two digits to the byte for the total of six bytes of storage.

ACTUAL Type	Definition
<i>Zn</i>	<p>Zoned decimal format (numeric string) where <i>n</i> is the number of digits (1 to 31), each of which takes one byte of storage. The last byte contains a digit and the sign.</p> <p>There are several standards for zoned data. For Read purposes, only right overpunched standards are supported and can be determined on Read since they are unique.</p> <p>The specific format to use when writing data must be known for read/write purposes. The default is ASCII right overpunch. To change the default, you must edit the EDACONF [.BIN]EDAENV.COM file and add the following logical:</p> <pre>DEFINE /NOLOG IBI_ZONED_OUT_TYPE {1 2}</pre> <p>where:</p> <p><u>1</u> Is the ASCII right overpunched standard (default).</p> <p><u>2</u> Is the EBCDIC right overpunched standard.</p> <p>Zoned right separate numeric or zoned left overpunched numeric formats are not supported.</p>
DATE	DATE indicates that the field is an OpenVMS 64-bit datetime stamp. This usage also requires an A4 filler field immediately following in the Master File.

The following conversions from ACTUAL format to USAGE (display) format are permitted:

ACTUAL	USAGE
A	A, D, F, I, P, date format
D	D
DATE	date format
F	F
I	I, date format
P	P, date format
Z	D, F, I, P

The following table shows the USAGE and ACTUAL formats for COBOL, FORTRAN, PL1, and Assembler field descriptions.

COBOL USAGE FORMAT	BYTES OF COBOL PICTURE	INTERNAL STORAGE	ACTUAL FORMAT	USAGE FORMAT
DISPLAY	X(4)	4	A4	A4
DISPLAY	S99	2	Z2	P3
DISPLAY	9(5)V9	6	Z6.1	P8.1
DISPLAY	99	2	A2	A2
COMP	S9	4	I2	I1
COMP	S9(4)	4	I2	I4
COMP*	S9(5)	4	I4	I5
COMP	S9(9)	4	I4	I9
COMP-1**	—	4	F4	F6
COMP-2***	—	8	D8	D15
COMP-3	9	8	P1	P1
COMP-3	S9V99	8	P2	P5.2
COMP-3	9(4)V9(3)	8	P4	P8.3
FIXED BINARY(7) (COMP-4)	B or XL1	8	I4	I7

*Equivalent to INTEGER in FORTRAN, FIXED BINARY(31) in PL/1, and F in Assembler.

**Equivalent to REAL in FORTRAN, FLOAT(6) in PL/1, and E in Assembler.

***Equivalent to DOUBLE PRECISION or REAL*8 in FORTRAN, FLOAT(16) in PL/1, and D in Assembler.

Note:

- The USAGE lengths shown are minimum values. They may be larger if desired. Additional edit options may also be added.
- In USAGE formats, an extra character position is required for the minus sign if negative values are expected.
- PICTURE clauses are not permitted for internal floating-point items.
- USAGE length should allow for maximum possible number of digits.
- In USAGE formats, an extra character position is required for the decimal point.

Describing Indexed Files (SUFFIX=RMS)

RMS keyed (indexed) files may be described using a SUFFIX of RMS, an attribute called GROUP, and a FIELDTYPE of I to designate a key. The GROUP attribute designates a key that is comprised of one or more contiguous fields. Additional keys may be described using either the GROUP or FIELD attribute, and a FIELDTYPE of I.

Segment Name for Indexed Files

The segment name (SEGNAME value) of the first segment in an indexed file (SUFFIX=RMS) must be ROOT. The remaining segments can have any valid segment name. The only exception to this rule is for unrelated record types where the first segment name value must be DUMMY.

Describing Keys

The primary key is defined by the GROUP attribute and an alias value of KEY in the Master File. If there is a secondary key consisting of more than one field, it must also be described by the GROUP attribute and a numbered key. Otherwise, the secondary key can be indicated by using a FIELDTYPE of I in the field description.

The primary key of an RMS file is defined using the GROUP attribute, consisting of one or more fields. A file might only have one keyfield, but it must still be described with the GROUP declaration. The GROUP must contain ALIAS=KEY. Coding KEY without ALIAS= is not sufficient. The GROUP declaration has the following syntax

```
GROUP=keyname, ALIAS=KEY, USAGE=usage, ACTUAL=actual,$
```

where:

keyname

Is a name of 1 to 66 characters.

The secondary keys of an RMS file are indicated by using ALIAS=KEY(n) and FIELDTYPE=I in the GROUP or FIELD declaration for the key. For example

```
GROUP=keyname, ALIAS=KEYn, USAGE=usage, ACTUAL=actual, FIELDTYPE=I,$
```

where:

keyname

Is a name of 1 to 48 characters.

KEYn

Indicates the alternate key.

The first alternate key is designated as KEY1, the second as KEY2, and so on.

Alternatively, if the secondary key is made up of only one field, the following syntax can be used:

```
FIELD=keyname, ALIAS=KEYn, USAGE=usage, ACTUAL=actual, FIELDTYPE=I,$
```


GROUP (for Contiguous Keys)

Contiguous keys consist of adjacent fields.

The GROUP attribute is used to define a contiguous primary key or a secondary key that is comprised of more than one field. The syntax is

```
GROUP=groupname , ALIAS=KEY[n] , USAGE=usage , ACTUAL=actual , FIELDTYPE=I , $
  FIELD=fieldname , ALIAS=alias , USAGE=usage , ACTUAL=actual , $
  .
  .
  .
  FIELD=fieldname , ALIAS=alias , USAGE=usage , ACTUAL=actual , $
```

where:

groupname

Is a name of 1 to 48 characters.

KEY[*n*]

Indicates a contiguous key. Use only KEY to specify a primary key. Use KEY[*n*] to specify a secondary key, where *n* is a number from 1 to 254 that indicates the key reference number.

FIELDTYPE=I

Makes the key accessible for reporting and indicates that the key can be used for direct retrieval. Do not use FIELDTYPE=I for a primary key.

usage

Is the data type and length designation.

actual

Is the data type and length designation.

For example, consider the contiguous key in the first part of the following Master File:

```
FILENAME=MANUALS , SUFFIX=RMS , $
SEGMENT=ROOT , SEGTYPE=S0 , $
GROUP=MDOCNUM      , ALIAS=KEY      , USAGE=A9      , ACTUAL=A9      , $
  FIELDNAME=DOCNUM  , ALIAS=DN      , USAGE=A5      , ACTUAL=A5      , $
  FIELDNAME=CODE    , ALIAS=CD      , USAGE=I6      , ACTUAL=I4      , $
  FIELDNAME=MRELEASE , ALIAS=MR      , USAGE=A7      , ACTUAL=A7      , $
  FIELDNAME=MPAGES  , ALIAS=MP      , USAGE=I5      , ACTUAL=I2      , $
  .
  .
  .
```

GROUP (for Discontiguous Keys)

Discontiguous keys consist of non-adjacent fields. If the GROUP attribute is used with discontiguous keys, the syntax is

```
GROUP=groupname ,ALIAS=DKEY[n] ,USAGE=usage ,ACTUAL=actual ,FIELDTYPE=I ,  
FIELD=           ,ALIAS=alias   ,USAGE=usage ,ACTUAL=actual ,  
.  
.  
.  
FIELD=           ,ALIAS=alias   ,USAGE=usage ,ACTUAL=actual ,
```

where:

DKEY[n]

Indicates that this GROUP is the key layout for a discontiguous key. The GROUP declaration must explicitly specify ALIAS=DKEY. Use only DKEY to specify a primary key. Use DKEY[*n*] to specify a secondary key, where *n* is a number from 1 to 254 that indicates the key reference number.

alias

Is the name of the base field to which the discontiguous key field corresponds. Note that the field name for the key field must be blank.

usage

Is the data type and length designation.

actual

Is the data type and length designation.

FIELDTYPE=I

Makes the key accessible for reporting and indicates that the key can be used for direct retrieval. Do not use FIELDTYPE=1 for a primary key.

The fields that comprise the discontiguous key are described in the order in which they appear in the record. They are then re-described using the GROUP attribute. The order within the GROUP is determined by their order of significance within the discontiguous key.

For example, consider the discontinuous keys in the first part of the following Master File:

```
FILENAME=MANUALS , SUFFIX=RMS , $
  SEGMENT=ROOT , SEGTYPE=S0 , $
    FIELDNAME=DOCNUM      , ALIAS=          , USAGE=A5      , ACTUAL=A5      , $
    FIELDNAME=CODE        , ALIAS=CD      , USAGE=I6      , ACTUAL=I4      , $
    FIELDNAME=MRELEASE    , ALIAS=        , USAGE=A7      , ACTUAL=A7      , $
    FIELDNAME=MPAGES      , ALIAS=        , USAGE=I5      , ACTUAL=I2      , $
  GROUP=MANUALS_KEY      , ALIAS=DKEY    , USAGE=A12     , ACTUAL=A12     , $
    FIELDNAME=            , ALIAS=DOCNUM  , USAGE=A5      , ACTUAL=A5      , $
    FIELDNAME=            , ALIAS=MRELEASE, USAGE=A7      , ACTUAL=A7      , $
    .
    .
    .
```

Note that within each segment, you can only have one instance of a key reference number. For example, consider the following: KEY5 and DKEY4 can be in the same segment because they do not reference the same key, but KEY4 and DKEY4 cannot be in the same segment because they reference the same key.

A GROUP of DKEY can occur only at the end of the segment, following all of the "real" field definitions since DKEY contains references to "real" fields as base fields.

Note: OCCURS segments cannot contain any key definitions. Non-OCCURS segments must contain either KEY or DKEY definitions.

USAGE and ACTUAL

For multi-field GROUPs, USAGE and ACTUAL formats are always alphanumeric. The ACTUAL attribute is A_n , where n is the sum of the actual lengths of the subordinate fields. The USAGE format is the sum of the internal storage lengths of the subordinate fields.

- Fields of USAGE I have an internal storage length of 4.
- Fields of USAGE F have an internal storage length of 4.
- Fields of USAGE P have an internal storage length of either 8 or 16.
- Fields of USAGE D have an internal storage length of 8.
- Alphanumeric fields have an internal storage length equal to the number of characters they contain as their field length. For example, fields of type A_n have an internal storage length of n .
- Natural date formats that are treated as integers have an internal storage length of 4.

For example, consider the discontinuous keys in a part of the following Master File:

```

.
.
.
FIELDNAME=FIELD1      , ALIAS=          , USAGE=P6      , ACTUAL=P2      , $
FIELDNAME=FIELD2      , ALIAS=          , USAGE=I9      , ACTUAL=I4      , $
FIELDNAME=FIELD3      , ALIAS=          , USAGE=A2      , ACTUAL=A2      , $
GROUP=ALTERNATE      , ALIAS=DKEY      , USAGE=A10     , ACTUAL=A4      , $
FIELDNAME=            , ALIAS=FIELD3    , USAGE=A2      , ACTUAL=A2      , $
FIELDNAME=            , ALIAS=FIELD1    , USAGE=P6      , ACTUAL=P2      , $
.
.
.

```

The GROUP declaration USAGE attribute tells how many positions to use for the group key. If this length is wrong, the group key will not be used correctly.

In this example, the lengths of the ACTUAL attributes for subordinate fields FIELD3 and FIELD1 total 4, which is the length of the ACTUAL attribute of the GROUP key. The lengths of the USAGE attributes for the subordinate fields total 8. However, the length of the GROUP key USAGE attribute is found by adding their internal storage lengths as specified by the field types: 2 for USAGE=A2 and 8 for USAGE=P6, for a total of 10.

Single-Field Secondary Keys

Single-field secondary keys must be described as fields with FIELDTYPE=I. The ALIAS must be the key reference number as described in the RMS File Description Language (FDL), that is, KEY n or DKEY n , where n is a number from 1 to 254. Secondary keys can be described as GROUPs if they consist of portions with dissimilar formats. For example,

```

FILENAME=CUST , SUFFIX=RMS , $
SEGNAME=ROOT , SEGTYPE=S0 , $
  GROUP=G      , ALIAS=KEY      , USAGE=A10     , ACTUAL=A10 , $
    FIELDNAME=SSN      , ALIAS=SSN      , USAGE=A10     , ACTUAL=A10 , $
    FIELDNAME=FNAME    , ALIAS=KEY1     , USAGE=A10     , ACTUAL=A10 , FIELDTYPE=I , $
    FIELDNAME=LNAME    , ALIAS=KEY2     , USAGE=A10     , ACTUAL=A10 , FIELDTYPE=I , $

```

Here, SSN is a primary key and FNAME and LNAME are secondary keys.

If you are not sure of the secondary keys associated with a given file, you can use the OpenVMS ANALYSE facility. Refer to the OpenVMS documentation for ANALYSE/RMS. The DIRECTORY command with the /FULL qualifier will also show how many keys the file has.

Describing Complex RMS Keyed

These topics discuss various ways of describing complex RMS keyed files.

Describing Multiple Record Types

Files may have records that must be deciphered according to a record type indicator in the record itself. Describing these files involves two things. First, each different record type will need its own segment. Second, the relationship between the different record types (that is, between the different segments) will need to be determined and expressed using the PARENT attribute for each segment.

Using RECTYPE

The syntax for defining a RECTYPE field is

```

FIELDNAME=RECTYPE, ALIAS=alias, USAGE={Pn}, ACTUAL=
[ ,ACCEPT=list/range ], $
{An} {An}
{In} {In}
{Fn} {Fn}
{Dn} {Dn}
{Zn} {Zn}

```

where:

RECTYPE

Is the required field name.

alias

Is the primary RECTYPE identifier. If there is an ACCEPT list or range, this value is any valid alias name.

list

Is a list of one or more lines of specific RECTYPE values for records that have the same segment layout. The maximum number of characters allowed in the list is 255. Each item in the list must be separated by either a blank or the keyword OR. If an item in the list contains embedded blanks or commas, it must be enclosed within single quotation marks ('). The list may contain a single RECTYPE value.

For example:

```

FIELDNAME=RECTYPE, ALIAS=A, USAGE=A1,
ACTUAL=A1, ACCEPT=A OR B OR C, $

```

range

Is a range of one or more lines of RECTYPE values for records that have the same segment layout. The maximum number of characters allowed in the range is 255. If either value contains embedded blanks or commas, it must be enclosed in single quotation marks (').

To specify a range of values, include the lowest value, the keyword TO, and the highest value, in that order.

For example:

```
FIELDNAME=RECTYPE , ALIAS=100 , USAGE=P3 , ACTUAL=P2 ,  
ACCEPT=70 TO 100 , $
```

When using an ACCEPT list or range, also called generalized RECTYPE, the RECTYPE field name is not unique across segments, therefore it must not be used in a WHERE clause of an SQL request. It is the responsibility of the client application to determine if the row for the RECTYPE is required. If the client application cannot do this, you must code individual segments for each RECTYPE.

To illustrate the use of the generalized RECTYPE capability in RMS file descriptions, consider the following record layouts in the DOC file. Record type DN is the root segment and contains the document number and title. Record types M, I, and C contain information about manuals, installation guides, and course guides, respectively. Notice that record types M and I have the same layout.

Record Type DN:

```
---KEY---  
+-----+  
DOCID  FILLER  RECTYPE  TITLE  
+-----+
```

Record Type M:

```
-----KEY-----  
+-----+  
MDOCID  MDATE  RECTYPE  MRELEASE  MPAGES  FILLER  
+-----+
```

Record Type I:

```
-----KEY-----  
+-----+  
IDOCID  IDATE  RECTYPE  IRELEASE  IPAGES  FILLER  
+-----+
```

Record Type C:

```

-----KEY-----
+-----+
CRSEDOC  CDATE  RECTYPE  COURSENUM  LEVEL  CPAGES  FILLER
+-----+

```

Without the ACCEPT attribute, each of the four record types must be described as separate segments in the Master File. For example, a unique set of field names must be provided for record type M and for record type I, although they have the same layout.

The generalized RECTYPE capability enables you to code just one set of field names that apply to the record layout for both record type M and record type I. The ACCEPT attribute can be used for any RECTYPE specification, even when there is only one acceptable value.

```

FILENAME=DOC, SUFFIX=RMS,$
SEGNAME=ROOT, SEGTYPE=S0,$
GROUP=DOCNUM, ALIAS=KEY, A5, A5, $
FIELD=DOCID, ALIAS=, A5, A5, $
FIELD=FILLER, ALIAS=, A5, A5, $
FIELD=RECTYPE, ALIAS=, A3, A3, $
FIELD=TITLE, ALIAS=, A18, A18,$
SEGNAME=MANUALS, PARENT=ROOT, SEGTYPE=S0, $
GROUP=MDOCNUM, ALIAS=KEY, A10, A10,$
FIELD=MDOCID, ALIAS=, A5, A5, $
FIELD=MDATE, ALIAS=, A5, A5, $
FIELD=RECTYPE, ALIAS=M, A3, A3, ACCEPT = M OR I,$
FIELD=MRELEASE, ALIAS=, A7, A7, $
FIELD=MPAGES, ALIAS=, I5, A5, $
FIELD=FILLER, ALIAS=, A6, A6, $
SEGNAME=COURSES, PARENT=ROOT, SEGTYPE=S0, $
GROUP=CRSEDOC, ALIAS=KEY, A10, A10,$
FIELD=CDOCID, ALIAS=, A5, A5, $
FIELD=CDATE, ALIAS=, A5, A5, $
FIELD=RECTYPE, ALIAS=C, A3, A3, $
FIELD=COURSENUM, ALIAS=, A4, A4, $
FIELD=LEVEL, ALIAS=, A2, A2, $
FIELD=CPAGES, ALIAS=, I5, A5, $
FIELD=FILLER, ALIAS=, A7, A7, $

```

Describing Related Record Types

Consider the LIBRARY file that contains three types of records, related by a combination of the key and the RECTYPE attribute. The ROOT records have a key that consists of the publisher's number. The BOOKINFO segment has a key that consists of that same publisher's number, plus a hard or soft-cover indicator. The SERIANO key consists of the first two elements, plus a record type.

In the sample file, the repetition of the publisher's number interrelates the three types of records. The Master File for this file would look like the following:

```
FILENAME=LIBRARY6 , SUFFIX=RMS , $
SEGNAME=ROOT , SEGTYPE=S0 , $
  GROUP=PUBKEY , ALIAS=KEY , USAGE=A10 , ACTUAL=A10 , $
  FIELDNAME=PUBNO , ALIAS=PN , USAGE=A10 , ACTUAL=A10 , $
  FIELDNAME=FILLER , ALIAS= , USAGE=A1 , ACTUAL=A1 , $
  FIELDNAME=RECTYPE , ALIAS=1 , USAGE=A1 , ACTUAL=A1 , $
  FIELDNAME=AUTHOR , ALIAS=AT , USAGE=A25 , ACTUAL=A25 , $
  FIELDNAME=TITLE , ALIAS=TL , USAGE=A50 , ACTUAL=A50 , $
SEGNAME=BOOKINFO , SEGTYPE=S0 , PARENT=ROOT , $
  GROUP=BOINKEY , ALIAS=KEY , USAGE=A11 , ACTUAL=A11 , $
  FIELDNAME=PUBNO1 , ALIAS=P1 , USAGE=A10 , ACTUAL=A10 , $
  FIELDNAME=BINDING , ALIAS=BI , USAGE=A1 , ACTUAL=A1 , $
  FIELDNAME=RECTYPE , ALIAS=2 , USAGE=A1 , ACTUAL=A1 , $
  FIELDNAME=PRICE , ALIAS=PR , USAGE=D8.2N , ACTUAL=D8 , $
SEGNAME=SERIANO , SEGTYPE=S0 , PARENT=BOOKINFO , $
  GROUP=SERIKEY , ALIAS=KEY , USAGE=A12 , ACTUAL=A12 , $
  FIELDNAME=PUBNO2 , ALIAS=P2 , USAGE=A10 , ACTUAL=A10 , $
  FIELDNAME=BINDING1 , ALIAS=B1 , USAGE=A1 , ACTUAL=A1 , $
  FIELDNAME=RECTYPE , ALIAS=3 , USAGE=A1 , ACTUAL=A1 , $
  FIELDNAME=SERIAL , ALIAS=SN , USAGE=A15 , ACTUAL=A15 , $
SEGNAME=SYNOPSIS , SEGTYPE=S0 , PARENT=ROOT , OCCURS=VARIABLE , $
  FIELDNAME=PLOTLINE , ALIAS=PLOTL , USAGE=A10 , ACTUAL=A10 , $
```

A typical query might request information on price and call numbers for a specific publisher's number:

```
PRINT PRICE AND SERIAL BY PUBNO
IF PUBNO EQ 1234567890 OR 9876054321
```

Since PUBNO is part of the key, the retrieval can be made and the processing continues. For greater speed retrieval, you could add search criteria based on the BINDING field, which is also part of the key.

Describing Unrelated Record Types

A file may contain records that are not related to each other. Records with varying RECTYPES exist independently of each other in a file, and the sequence of records in the file may be random.

Consider our LIBRARY file. Suppose that the file has three types of records: book information, magazine information, and newspaper information.

Since book information, magazine information, and newspaper information have nothing in common, these record types cannot be described in a parent/child relationship.

The records simply look like the following:

```
BOOK      MAGAZINE    NEWSPAPER
```

To describe a file with unrelated records, you must make the record types descendants of a root segment named DUMMY.

The following rules apply to the DUMMY segment:

- The root (top) segment name must be DUMMY.
- It can have only one field, with an empty FIELDNAME and ALIAS.
- Both its USAGE and ACTUAL attributes must be A1.

All of the other segments must be descendants of the DUMMY segment.

The Master File for this file would look like the following:

```

FILENAME=LIBRARY3 , SUFFIX=FIX , $
SEGMENT=DUMMY , SEGTYPE=S0 , $
    FIELDNAME=          , ALIAS=          , USAGE=A1          , ACTUAL=A1          , $
SEGMENT=BOOK , SEGTYPE=S0 , PARENT=DUMMY , $
    FIELDNAME=RECTYPE   , ALIAS=B          , USAGE=A1          , ACTUAL=A1          , $
    GROUP=PUBNUM        , ALIAS=KEY        , USAGE=A10         , ACTUAL=A10         , $
    FIELDNAME=PUBNO     , ALIAS=PN         , USAGE=A10         , ACTUAL=A10         , $
    FIELDNAME=AUTHOR    , ALIAS=AT         , USAGE=A25         , ACTUAL=A25         , $
    FIELDNAME=TITLE     , ALIAS=TL         , USAGE=A50         , ACTUAL=A50         , $
    FIELDNAME=BINDING   , ALIAS=BI         , USAGE=A1          , ACTUAL=A1          , $
    FIELDNAME=PRICE     , ALIAS=PR         , USAGE=D8 . 2N     , ACTUAL=D8          , $
    FIELDNAME=SERIAL    , ALIAS=SN         , USAGE=A15         , ACTUAL=A15         , $
    FIELDNAME=SYNOPSIS  , ALIAS=SY         , USAGE=A150        , ACTUAL=A150        , $
SEGMENT=MAGAZINE , SEGTYPE=S0 , PARENT=DUMMY , $
    FIELDNAME=RECTYPE   , ALIAS=M          , USAGE=A1          , ACTUAL=A1          , $
    GROUP=PERNUM        , ALIAS=KEY        , USAGE=A10         , ACTUAL=A10         , $
    FIELDNAME=PER_NO    , ALIAS=PN         , USAGE=A10         , ACTUAL=A10         , $
    FIELDNAME=PER_NAME  , ALIAS=NA         , USAGE=A50         , ACTUAL=A50         , $
    FIELDNAME=VOL_NO    , ALIAS=VN         , USAGE=I2          , ACTUAL=I2          , $
    FIELDNAME=ISSUE_NO  , ALIAS=IN         , USAGE=I2          , ACTUAL=I2          , $
    FIELDNAME=PER_DATE  , ALIAS=DT         , USAGE=YYMD        , ACTUAL=DATE        , $
    FIELDNAME=          , ALIAS=FILLER    , USAGE=A4          , ACTUAL=A4          , $
SEGMENT=NEWSPAP , SEGTYPE=S0 , PARENT=DUMMY , $
    FIELDNAME=RECTYPE   , ALIAS=N          , USAGE=A1          , ACTUAL=A1          , $
    FIELDNAME=NEW_NAME  , ALIAS=NN         , USAGE=A50         , ACTUAL=A50         , $
    FIELDNAME=NEW_DATE  , ALIAS=ND         , USAGE=I6MDY       , ACTUAL=I4          , $
    FIELDNAME=          , ALIAS=FILLER    , USAGE=A4          , ACTUAL=A4          , $
    GROUP=PAPVI         , ALIAS=KEY        , USAGE=A4          , ACTUAL=A4          , $
    FIELDNAME=NVOL_NO   , ALIAS=NV         , USAGE=I2          , ACTUAL=I2          , $
    FIELDNAME=NISSUE    , ALIAS=NI         , USAGE=I2          , ACTUAL=I2          , $

```

Describing Embedded Repeating Data

Some records may contain embedded repeating data. Consider the following record layout:

A B C1 C2 C1 C2

Fields C1 and C2 repeat within this data record. C1 has an initial value, as does C2. C1 then provides a second value for that field, as does C2. Thus, there are two values for fields C1 and C2 for every one value for fields A and B.

The number of times C1 and C2 occur does not have to be fixed, depending on the value of a counter field. Suppose field B is this counter field. In the case shown above, the value of field B is 2, since C1 and C2 occur twice. The value of field B in the next record may be different, and fields C1 and C2 will occur that number of times.

The number of times fields C1 and C2 occur can also be variable. In this case, everything after fields A and B is assumed to be a series of C1s and C2s, alternating to the end of the record.

You describe these multiply occurring fields by placing them in a separate segment. Fields A and B are placed in the first segment, called the root segment. Fields C1 and C2, which occur multiple times in relation to A and B, are placed in a descendant segment. You use an additional segment attribute, the OCCURS attribute, to specify that this segment is a multiply occurring segment.

Repeating fields or groups of fields described by the OCCURS attribute are not supported for free-format (comma-delimited) sequential files.

OCCURS

The OCCURS attribute is an optional segment attribute used to describe records containing repeating fields or groups of fields. You define such records by describing the singly occurring fields in one segment and the multiply occurring fields in another, subordinate segment. The OCCURS attribute appears in the declaration for the subordinate segment.

The syntax is

```
OCCURS = {n | fieldname | VARIABLE},$
```

where:

n

Is an integer value showing the number of occurrences (1 to 4095).

fieldname

Names a data field in the parent segment, that is, a counter specifying the number of occurrences of the descendant segment.

VARIABLE

Indicates that the number of occurrences varies from record to record. The number of occurrences is computed from the record length (that is, if the field lengths for the segment add up to 40, and 120 characters are read in, it means there are three occurrences).

When different types of records are combined in one file, each record type can contain only one segment (defined as OCCURS=VARIABLE). It may have OCCURS descendants (if it contains a nested group), but it may not be followed by any other segment with the same parent, that is, there can be no other segments to its right in the structure. This restriction is necessary to ensure that data in the record is interpreted correctly.

Example Using the OCCURS Attribute

You place the OCCURS attribute in the segment declaration after the PARENT attribute. Consider the following record layout:

```
A   B   C1   C2   C1   C2
```

You have two occurrences of fields C1 and C2 for every one occurrence of fields A and B. Thus, to describe this file, you place fields A and B in the root segment, and fields C1 and C2 in the descendant segment.

You describe this file with the following Master File:

```
FILENAME=EXAMPLE1 , SUFFIX=RMS , $
GROUP=ONE , ALIAS=KEY , USAGE=A2 , ACTUAL=A2 , FIELDTYPE=I , $
FIELDNAME=A , ALIAS= , USAGE=A2 , ACTUAL=A2 , $
FIELDNAME=B , ALIAS= , USAGE=A1 , ACTUAL=A1 , $
SEGNAME=TWO , PARENT=ONE , OCCURS=2 , SEGTYPE=S0 , $
FIELDNAME=C1 , ALIAS= , USAGE=I4 , ACTUAL=I2 , $
FIELDNAME=C2 , ALIAS= , USAGE=I4 , ACTUAL=I2 , $
```

Note: OCCURS is not supported for Write Access. However, if the fields occur a specific number of times, an alternate Master File can be built with the fields described that number of times (for example, PAYMENT_1, PAYMENT_2 if Payment occurs two times). Using the alternate Master, specific instances of the OCCURS can be referenced. If the OCCURS segments will not be referenced during the write, the alternate Master is not needed. However, the message (FOC1305) RMS Duplicate Record will display.

Example Describing Parallel and Nested Sets of OCCURS Segments

You can have several sets of repeating fields in your data structure. You describe each of these sets of fields as a separate segment in your Master File.

Sets of repeating fields can be divided into two basic types: parallel and nested. Parallel sets of repeating fields are unrelated (that is, they have no parent/child or logical relationship). Consider the following record layout:

A1 A2 B1 B2 B1 B2 C1 C2 C1 C2 C1 C2

In this example, fields B1 and B2 and fields C1 and C2 repeat within the record. The number of times that fields B1 and B2 occur is unrelated to the number of times fields C1 and C2 occur. Fields B1 and B2 and fields C1 and C2 are parallel sets of repeating fields. They should be described in the Master File as children of the same parent, the segment that contains fields A1 and A2. The following Master File illustrates this relationship:

```
FILENAME=EXAMPLE1, SUFFIX=RMS,$
GROUP=ONE, ALIAS=KEY, USAGE=A2, ACTUAL=A2, FIELDTYPE=I, $
FIELDNAME=A1, ALIAS=, USAGE=A2, ACTUAL=A2, $
FIELDNAME=A2, ALIAS=, USAGE=A2, ACTUAL=A2, $
SEGNAME=TWO, SEGTYPE=S0, PARENT=ONE, OCCURS=2, $
FIELDNAME=B1, ALIAS=, USAGE=A2, ACTUAL=A2, $
FIELDNAME=B2, ALIAS=, USAGE=A2, ACTUAL=A2, $
SEGNAME=THREE, SEGTYPE=S0, PARENT=ONE, OCCURS=3, $
FIELDNAME=C1, ALIAS=, USAGE=A2, ACTUAL=A2, $
FIELDNAME=C2, ALIAS=, USAGE=A2, ACTUAL=A2, $
```

Nested sets of repeating fields are those whose occurrence in some way depends on one another. Consider the following data structure:

A1 A2 B1 B2 C1 C1 B1 B2 C1 C1 C1

In this example, field C1 occurs after fields B1 and B2 occur once. Field C1 occurs a varying number of times, as recorded by a counter field, B2. There will not be a set of occurrences of C1 unless C1 is preceded by an occurrence of fields B1 and B2. Fields B1, B2, and C1 are a nested set of repeating fields. The following Master File illustrates this relationship:

```
FILENAME=EXAMPLE2, SUFFIX=RMS,$
GROUP=ONE, ALIAS=KEY, USAGE=A2, ACTUAL=A2, FIELDTYPE=I, $
FIELDNAME=A1, ALIAS=, USAGE=A2, ACTUAL=A2, $
FIELDNAME=A2, ALIAS=, USAGE=A2, ACTUAL=A2, $
SEGNAME=TWO, SEGTYPE=S0, PARENT=ONE, OCCURS=2, $
FIELDNAME=B1, ALIAS=, USAGE=A2, ACTUAL=A2, $
FIELDNAME=B2, ALIAS=, USAGE=I6, ACTUAL=I4, $
SEGNAME=THREE, SEGTYPE=S0, PARENT=TWO, OCCURS=B2, $
FIELDNAME=C1, ALIAS=, USAGE=A2, ACTUAL=A2, $
```

Since field C1 repeats with relation to fields B1 and B2, which repeat in relation to fields A1 and A2, field C1 is described as a separate, descendant segment of Segment TWO, which is in turn a descendant of Segment ONE.

The following data structure contains both nested and parallel sets of repeating fields:

A1 A2 B1 B2 C1 C1 C1 B1 B2 C1 C1 C1 C1 D1 D1 E1 E1 E1 E1

You describe this with the Master File that follows. The PARENT attribute is used to describe the logical relationship of the segments. Note that the assignment of the PARENT attribute shows you how the occurrences are nested:

```
FILENAME=EXAMPLE3 , SUFFIX=RMS , $
GROUP=ONE , ALIAS=KEY , USAGE=A2 , ACTUAL=A2 , FIELDTYPE=I , $
  FIELDNAME=A1 , ALIAS= , USAGE=A2 , ACTUAL=A2 , $
  FIELDNAME=A2 , ALIAS= , USAGE=I4 , ACTUAL=I1 , $
SEGNAME=TWO , PARENT=ONE , OCCURS=2 , SEGTYPE=S0 , $
  FIELDNAME=B1 , ALIAS= , USAGE=A15 , ACTUAL=A15 , $
  FIELDNAME=B2 , ALIAS= , USAGE=I4 , ACTUAL=I1 , $
SEGNAME=THREE , PARENT=TWO , OCCURS=B2 , SEGTYPE=S0 , $
  FIELDNAME=C1 , ALIAS= , USAGE=A25 , ACTUAL=A25 , $
SEGNAME=FOUR , PARENT=ONE , OCCURS=A2 , SEGTYPE=S0 , $
  FIELDNAME=D1 , ALIAS= , USAGE=A15 , ACTUAL=A15 , $
SEGNAME=FIVE , PARENT=ONE , OCCURS=VARIABLE , SEGTYPE=S0 , $
  FIELDNAME=E1 , ALIAS= , USAGE=A5 , ACTUAL=A5 , $
```

Note:

- Segments ONE, TWO, and THREE represent a nested group of repeating segments. Fields B1 and B2 occur a fixed number of times, so OCCURS equals 2. Field C1 occurs a certain number of times within each occurrence of fields B1 and B2. The number of times C1 occurs is determined by the value of B2, which is a counter. The value of B2 is 3 for the first occurrence of Segment TWO, and 4 for the second occurrence.
- Segments FOUR and FIVE consist of fields that repeat independently within the parent segment. They have no relationship to each other or to Segment TWO except for the common parent, so they represent a parallel group of repeating segments.
- As in the case of Segment THREE, the number of times Segment FOUR occurs is determined by a counter in its parent, A2. In this example, the value of A2 is 2.
- The number of times Segment FIVE occurs is VARIABLE. The rest of the fields in the record (all those to the right of the first occurrence of E1) as occurrences of E1. To ensure that data in the record is interpreted correctly, a segment defined as OCCURS=VARIABLE must be at the end of the record. In a data structure diagram, it will be the right-most segment. There can only be one segment defined as OCCURS=VARIABLE for each parent.

POSITION

The POSITION attribute is an optional attribute. It is used to describe a record in which non-repeating data follows embedded repeating data.

You describe the file as a multi-segment structure, made up of a parent segment and at least one child segment that contains the embedded repeating data. The parent segment is made up of whatever singly occurring fields are in the record, as well as an alphanumeric field (or fields) that indicates where the embedded repeating data appear in the record. The alphanumeric field is a placeholder that is the exact length of the combined embedded repeating data. For example, if you have four occurrences of an 8-character field, the length of the placeholder in the parent segment will be 32 characters.

The POSITION attribute is described in the child segment. It gives the name of the placeholder field in the parent segment.

The syntax is

```
POSITION = fieldname
```

where:

```
fieldname
```

Is the name of the field in the parent segment that defines the starting position of the multiply occurring fields.

Consider the following record layout:

```
A1  Q1  Q1  Q1  Q1  A2  A3  A4
```

In this example, field Q1 repeats four times in the middle of the record. When you describe this structure, you specify a field that occupies the position of the four Q1 fields in the record. You then assign the actual Q1 fields to a descendant, multiply occurring segment. The POSITION attribute, specified in the descendant segment, gives the name of the placeholder field in the parent segment.

The Master File for this file would look like this:

```
FILENAME=EXAMPLE3 , SUFFIX=RMS , $
GROUP=ONE , ALIAS=KEY , USAGE=A2 , ACTUAL=A2 , FIELDTYPE=I , $
FIELDNAME=A1 , ALIAS= , USAGE=A2 , ACTUAL=A2 , $
FIELDNAME=QFIL , ALIAS= , USAGE=A1 , ACTUAL=A32 , $
FIELDNAME=A2 , ALIAS= , USAGE=I2 , ACTUAL=I2 , $
FIELDNAME=A3 , ALIAS= , USAGE=A10 , ACTUAL=A10 , $
FIELDNAME=A4 , ALIAS= , USAGE=A15 , ACTUAL=A15 , $
SEGNAME=TWO , SEGTYPE=S0 , PARENT=ONE , POSITION=QFIL , OCCURS=4 , $
FIELDNAME=Q1 , ALIAS= , USAGE=D8 , ACTUAL=D8 , $
```


If the total length of the multiple occurrences of the field(s) is greater than 256, you can use a filler field after the place holder field to make up the remaining length. This is because the format of a field cannot exceed 256 bytes.

This structure will only work if you have a fixed number of occurrences of the repeating field. This means that the OCCURS attribute of the descendant segment must be of the type OCCURS=*n*. The following will not work: OCCURS=*fieldname* or OCCURS=VARIABLE.

When a segment is coded with ...OCCURS=*n*, POSITION=*fieldname*, ..., all segments that follow using OCCURS=*n* or OCCURS=*fieldname*, must use the POSITION attribute to correctly map data.

ORDER Field

In an OCCURS segment, the order of the data may be significant. For example, the numbers may represent monthly or quarterly data, but the record itself may not explicitly specify the month (or quarter) to which the data applies.

You can add a special ORDER field to your Master File to identify the individual occurrences of data uniquely within their parent segment. This is typically done to the screen by the occurrence number versus a field value (for example, the month is 12). The sequence number of each instance of the segment is automatically defined as a virtual field, and does not actually exist within the file.

The following syntax rules apply to the ORDER field:

- It must be the last field described in an OCCURS segment. If you are using MAPVALUE, then MAPVALUE must be the last field preceded by the ORDER field.
- The field name is arbitrary.
- The ALIAS attribute must be ORDER.
- The USAGE attribute must be *In*, with any appropriate edit options; "*n*" is a number from 1 to 9.
- The ACTUAL attribute must be *I4*.

For example:

```
FIELD=ACT_MONTH, ALIAS=ORDER, USAGE=I2, ACTUAL=I4,$
```

Order values are assigned sequentially (1,2,3,...) and the order value is reset to 1 when a new instance of the parent segment is retrieved. The value is assigned prior to any selection tests that might accept or reject the record, and can be used in a screening condition. For example, to obtain data for the month of June, type:

```
SUM AMOUNT...
IF ORDER IS 6
```

Example Describing Repeating Embedded Data With Record Type Indicators

If a file contains records that have repeating embedded data, the OCCURS attribute is used to describe a separate segment for the repeating fields. In some files, however, the repeating fields themselves must be identified according to a record type indicator. Suppose you want to describe a file that, schematically, looks like the following:

```
A  Rectype B C      Rectype B C
A  Rectype D        Rectype D
```

You would need to describe three segments in your Master File, with A as the root segment, and segments for B, C, and D, as two descendant OCCURS segments for A.

This is its Master File:

```
FILE=ABCD, SUFFIX=RMS,$
GROUP=ONE ,ALIAS=KEY ,ACTUAL=A2 ,USAGE=A2 ,,$
FIELDNAME=A ,ALIAS= ,USAGE=A2 ,ACTUAL=A2 ,,$
SEGNAME=BC_SEG, SEGTYPE=S0, PARENT=ONE, OCCURS=2,$
FIELDNAME=RECTYPE ,ALIAS=1 ,USAGE=I2 ,ACTUAL=I2 ,,$
FIELDNAME=B ,ALIAS= ,USAGE=A5 ,ACTUAL=A5 ,,$
FIELDNAME=C ,ALIAS= ,USAGE=A5 ,ACTUAL=A5 ,,$
SEGNAME=D_SEG, SEGTYPE=S0, PARENT=ONE, OCCURS=2,$
FIELDNAME=RECTYPE ,ALIAS=2 ,USAGE=I2 ,ACTUAL=I2 ,,$
FIELDNAME=D ,ALIAS= ,USAGE=A15 ,ACTUAL=A15 ,,$
```

Each of the two descendant OCCURS segments in this example depends on the record type indicator that appears for each occurrence.

All the rules of syntax for using RECTYPE fields and OCCURS segments apply to RECTYPES within OCCURS segments.

Since the OCCURS segment depends on the RECTYPE indicator for its evaluation, the RECTYPE must appear at the start of each OCCURS segment. This allows very complex files to be described, including those with nested and parallel repeating groups that depend on RECTYPES. For example:

```
A  Rectype B C  Rectype D  Rectype D  Rectype E  Rectype E
```

In this case, B/C, and D represent a nested repeating group, and E represents a parallel repeating group.

Syntax**How to Describe Repeating Groups Depending on a Preceding Record Type Indicator Using MAPFIELD/MAPVALUE**

MAPFIELD is assigned as the ALIAS of the field that will be the record type indicator. You can give this field any name; it is otherwise described according to the usual syntax:

```
FIELD=name, ALIAS=MAPFIELD, USAGE=usage, ACTUAL=actual,$
```

The descendant segments, whose values depend on the value of the MAPFIELD, are described as separate segments, one for each possible value of MAPFIELD, and all descending from the segment that has the MAPFIELD. A special field, MAPVALUE, is described as the last field in these descendant segments. If an ORDER field is chosen, it must be used before the MAPVALUE. The actual MAPFIELD value is supplied as the MAPVALUE's ALIAS.

The syntax is

```
FIELDNAME=MAPVALUE, ALIAS=alias, USAGE={Fn} ACTUAL={Fn} ,
ACCEPT=list/range], $
                                {Dn},      {Dn}
```

where:

MAPVALUE

Indicates that the segment depends on a MAPFIELD in its parent segment.

alias

Is the primary MAPFIELD value that identifies the segment. If there is an ACCEPT list, this value is any value in the ACCEPT list or range.

USAGE

Is the same format as the MAPFIELD format in the parent segment.

ACTUAL

Is the same format as the MAPFIELD format in the parent segment.

list

Is a list of one or more lines of specific MAPFIELD values for records that have the same segment layout. The maximum number of characters allowed in the list is 255. Each item in the list must be separated by either a blank or the keyword OR. If an item in the list contains embedded blanks or commas, it must be enclosed within single quotation marks ('). The list may contain a single MAPFIELD value. For

```
FIELDNAME=MAPVALUE, ALIAS=A, USAGE=A1,
ACTUAL=A1, ACCEPT=A OR B OR C, $
```

range

Is a range of one or more lines of MAPFIELD values for records that have the same segment layout. The maximum number of characters allowed in the range is 255. If either value in the range contains embedded blanks or commas, it must be enclosed in single quotation marks (').

To specify a range of values, include the lowest value, the keyword TO, and the highest value, in that order.

In some cases, the record type indicates what kind of repeating data will follow. Schematically, the records would look like the following:

```
A B Recordtype (1) C D C D C D
A B Recordtype (2) E E
```

The first record contains "header" information, values for A and B, followed by an OCCURS segment of C and D that was identified by its preceding record type indicator. The second record has a different record type indicator and contains a different repeating group, this time for E.

Since the OCCURS segments are identified by the record type indicator, rather than the parent A/B segment, you can use the keyword MAPFIELD. MAPFIELD identifies a field like RECTYPE, but, since the OCCURS segments will each have specific values for MAPFIELD, its value is associated with each OCCURS segment by means of a complementary field named MAPVALUE.

The Master File for this file would look like the following:

```
FILENAME=EXAMPLE , SUFFIX=RMS , $
GROUP=ONE , ALIAS=KEY , ACTUAL=A2 , USAGE=A2 , $
FIELDNAME=A , ALIAS= , USAGE=A2 , ACTUAL=A2 , $
FIELDNAME=B , ALIAS= , USAGE=A10 , ACTUAL=A10 , $
FIELDNAME=FLAG , ALIAS= MAPFIELD , USAGE=A1 , ACTUAL=A1 , $
SEGNAME=TWO , SEGTYPE=S0 , PARENT=ONE , OCCURS=VARIABLE , $
FIELDNAME=C , ALIAS= , USAGE=A5 , ACTUAL=A5 , $
FIELDNAME=D , ALIAS= , USAGE=A7 , ACTUAL=A7 , $
FIELDNAME=MAPVALUE , ALIAS=1 , USAGE=A1 , ACTUAL=A1 , $
SEGNAME=THREE , SEGTYPE=S0 , PARENT=ONE , OCCURS=VARIABLE , $
FIELDNAME=E , ALIAS= , USAGE=D12.2 , ACTUAL=D8 , $
FIELDNAME=MAPVALUE , ALIAS=2 , USAGE=A1 , ACTUAL=A1 , $
```

Assigning a Keyed RMS File to a Master File

In order to associate a keyed RMS file to a Master File, an Access File must exist.

The Access File will have the following format

```
RMSFILE = filename,
[ACCESS = {SHARED|READONLY},]
[LOCKMODE = {SKIP|KEEP} ,]
[STATMODE = {EXCEPTIONS|ON|OFF},] $
```

where:

filename

Is one of the following:

1. A fully qualified file name, including files on remote hosts.
2. A logical that points to a local or fully qualified RMS file.

The Access File is also used to specify ACCESS, LOCKMODE, and STATMODE for RMS files.

- ACCESS can have a value of SHARED for read/write access, or READONLY for Read access.
- LOCKMODE is for retrieval only and can have a value of SKIP or KEEP. If you select SKIP, the Data Adapter for RMS will skip locked records. If you select KEEP, the Data Adapter for RMS will retrieve locked records.
- STATMODE can have a value of EXCEPTIONS, ON, or OFF. EXCEPTIONS is the default and will display a message when locked records are encountered. ON allows a message to occur (whether or not locked records are encountered), and OFF suppresses locking messages.

See *File and Record Locking* on page 32-34 for more information on ACCESS, LOCKMODE, and STATMODE.

You can override the RMSFILE declaration in the Access File with a FILEDEF. The FILEDEF can be done locally within a procedure (before accessing the file) or within the profile. For example,

```
FILEDEF rmsfile DISK filename
```

where:

rmsfile

Is the logical metadata reference name matching the physical file name.

filename

Is the OpenVMS file name. It may be any valid OpenVMS file specification or logical name pointing to a file.

File and Record Locking

This topic describes features that enable a developer to control file and record locking. Lock conflict, or contention, can occur at file level or record level. File contention is caused by incompatible access by two or more processes. Record contention is caused when a requested lock is incompatible with any existing locks on the record.

The following terms summarize the types of access allowed to RMS files and records:

Read Access	Provides users with access to a file and the records in the file for actions that make no alterations. Read can use any access mode (dependent on site needs) for the actions that are defined in the following tables.
Read/Write Access	Provides users with access to a file and the records in the file for actions that make alterations to records, add records, or delete records. Read/write access requires the use of SHARED as the access mode. All other modes are not supported WRITE operations.

File Locking and the Interface to RMS

When your request attempts to open an RMS file, the Access Parameter within the Access File determines what you can do at a file level with the RMS file, and what actions others can take while you have the file open.

When a request attempts to open an RMS file, the Access Parameter determines which type of access to use. If the Access Parameter specification is incompatible with another process' access to the file, the procedure fails because it is not able to open the file. Additionally, if the file is opened for read, but a subsequent write is attempted, it will fail due to an inappropriate access mode.

Depending on how a file is opened, as specified by the Access Parameter, subsequent opens are limited as follows:

		Processes by Others			
		READONLY	SHARED	PROTECTED	EXCLUSIVE
Processes by Server Users	READONLY	Allowed	Allowed	Allowed	Denied
	SHARED	Allowed	Allowed	Allowed when SHARED process is doing read-access operation.	Denied

This table describes user access and file sharing options used in response to the following Access File options for the Access Parameter.

Access Parameter	User Access	File Sharing to Other Processes
READONLY	Read	Read and Write
SHARED	Read and Write	Read and Write

Record Locking

RMS requests are no-lock for each record retrieved for READ operations.

Handling Locked Records During Table Read Request

RMS files can be accessed while they are simultaneously accessed by other programs. For instance, the file might be in use by another program that is maintaining it by adding, deleting, or updating records.

When a READ operation is rejected due to a lock conflict by another process, aborts the request and displays informative messages. You can override this behavior on a file-by-file basis using the LOCKMODE and STATMODE options in the Access File. LOCKMODE and STATMODE do not apply to WRITE requests such as UPDATE or DELETE.

LOCKMODE

LOCKMODE can affect record retrieval in one of two ways. Each of these settings and the affect they have on the retrieval process is as follows:

- Omitting LOCKMODE from the acx file.
To halt retrieval when locked records are encountered, omit LOCKMODE from the acx file. A (FOC1325) RMS READ LOCK ABORT ERROR will be sent if locked records are encountered during record retrieval. Records retrieved before the lock is encountered may have already been sent to the client and should be discarded.
- Setting one of two acceptable values in the acx file.
- To retrieve all records, including locked records, code LOCKMODE=KEEP in the acx file. For example:
`LOCKMODE = KEEP`
- To retrieve all records, except locked records, and complete the request, code LOCKMODE=SKIP in the acx file. For example:
`LOCKMODE = SKIP`

STATMODE

STATMODE controls whether or not a message is sent to the client program about retrieved records. STATMODE has three settings:

- ON

To receive a message giving the statistics of data retrieval, set STATMODE to ON in the Access File. The message displays, in addition to the report request, after data retrieval is complete. For example:

```
STATMODE = ON
```

A FOC1320 message displays when no locked records are encountered. For example:

```
(FOC1320) RMS STATS  :(SEGCAR    ) Reads = 1, Skips = 0, Keeps = 0
```

If LOCKMODE is set to KEEP a FOC1324 message displays. For example:

```
(FOC1324) RMS KEEP   :(SEGCAR    ) Reads = 1, Skips = 2, Keeps = 0
```

If LOCKMODE is set to SKIP, a FOC1322 message displays. For example:

```
(FOC1322) RMS SKIP   :(SEGCAR    ) Reads = 1, Skips = 0, Keeps = 2
```

- OFF

Set the STATMODE to OFF if no messages are to be sent when locked records are encountered. The client application will not have any indication whether or not locked records were encountered during record retrieval.

- EXCEPTIONS

To receive a message giving the statistics of data retrieval, only if locked records are encountered, set the STATMODE to EXCEPTIONS. The message displays, in addition to the report request, after data retrieval is complete. No message displays if no locked records are encountered during retrieval. For example:

```
STATMODE = EXCEPTIONS
```

If LOCKMODE is set to KEEP a FOC1324 message displays. For example:

```
(FOC1324) RMS KEEP   :(SEGCAR    ) Reads = 1, Skips = 2, Keeps = 0
```

If LOCKMODE is set to SKIP, in addition to the report output, a FOC1322 message displays. For example:

```
(FOC1322) RMS SKIP   :(SEGCAR    ) Reads = 1, Skips = 0, Keeps = 2
```


Access File Examples

The following is an example of a Remote Node:

```
RMSFILE=HOST1 : : DISK$PROG : [ PROD . INFO ] EMPINFO . DAT ,
ACCESS=SHARED , $
```

The following is an example of a Full Path as defined by a logical and a file name:

```
RMSFILE=DATADIR : PERSON . DAT , ACCESS=READONLY , $
```

The following is an example of a Logical Name:

```
RMSFILE=MYLOGICAL , ACCESS=SHARED , $
```

The following is an example of a Full Path on a file using LOCKMODE and STATMODE:

```
RMSFILE=DISK100 : [ DATA ] PERSON . DAT ,
ACCESS=READONLY , LOCKMODE=SKIP , STATMODE=OFF , $
```

Retrieving Data From RMS Files

The following topics outline the procedures necessary for retrieving data from RMS files.

Index Selection

By default, the primary key is used for retrieval of records from indexed RMS files. You can override this default with the Automatic Index Selection (AIS).

The primary benefit of keys is improved efficiency in record retrieval. They provide an alternate, more efficient, retrieval method and can be used with screening tests on the selected key which are translated into direct reads. Indexes can also be used to control the order of retrieval of records.

To screen more than one field when working with group keys that are comprised of multiple fields, use a slash (/) character to separate the components of the group key. Screening is limited to NE, IS, and EQ relationships. The syntax is:

```
... groupname {NE} [' ]value/value.../value[' ]
           {IS}
           {EQ}
```

where:

groupname

Is the GROUP field name in the Master File.

The slash (/) separates the parts of the group field. If you omit the optional quotation marks, embedded blanks in the value will be removed. Use single quotations marks (') around the values to preserve the values exactly as typed.

Automatic Index Selection (AIS)

The Automatic Index Selection (AIS) facility automatically selects a key for direct access to an indexed file based on a request.

Requirements for Using AIS

AIS automatically uses a key for direct retrieval when an applicable screening condition is reached in a request. AIS is used when all of the following requirements are met:

- Primary keys are described in a Master File with the ALIAS=KEY or ALIAS=DKEY attribute placed on the GROUP declaration (not on every field in the GROUP). Note that the primary key must always be described as a GROUP.
- Secondary keys are described with the FIELDTYPE attribute on the GROUP or on the FIELD declaration for a secondary key not belonging to a group.
- There is an optimizable screening condition on a key field. If a key is defined as a GROUP, the screening condition must be on either the GROUP name, or on the first field of the key. Only those key fields in a screening condition with one of the following logical relations are used for index selection:
 - Equal
 - Less than or equal to
 - Less than
 - Greater than
 - Greater than or equal to
- AIS applies only to keys in the root segment.

Syntax for RMS Master File Attributes

This topic details the syntax for the attributes used to describe files, segments, and fields.

File Attributes

File Declaration:

```
FILE[NAME]=filename, [FILE]SUFFIX=type , $
```

Attribute	Function	Value
FILE[NAME]	Identifies the Master File.	All characters and digits, but no embedded blanks. Eight characters maximum.
[FILE]SUFFIX	Identifies the type of file to which the Master File applies.	RMS.

Note: The SUFFIX values listed here pertain to the RMS files; the SUFFIX parameter can have other values for data that resides on other Database Management Systems that can be accessed. Consult the appropriate Data Adapter manuals for these products.

Except for the following differences, the description of data and relationships between fields within a file is the same for keyed (indexed) RMS files, FIX (fixed-format sequential) files, and COM (comma-delimited) files. Keyed (indexed) RMS uses the RMS File System to access data using information in the Access File declarations. The others use sequential access using a FILEDEF to identify and access the file. Index related declarations do not apply to fixed-sequential or comma-delimited files.

Segment Attributes

Segment Declaration:

```
{SEGNAME | SEGMENT} = segname [ , SEGTYPE=S0 ] [ , PARENT=parent_name ]
[ {n } ]
[ , OCCURS= {fieldname} ] [ , POSITION=fieldname ], $
[ {VARIABLE } ]
```

Attribute	Function	Value
SEGNAME SEGMENT	Identifies a collection of data fields that are related.	All characters and digits, except special characters. Up to eight characters in length.
SEGTYPE	Identifies the physical storage of the segment.	S0.
PARENT	Identifies the "parent" or owner of the current segment.	Any valid segment name previously defined in the Master File.
OCCURS	Identifies the field as a multiply-occurring field and specifies how the number of occurrences will be determined.	Any integer from 1 to 4095 or any valid field name or VARIABLE.
POSITION	Identifies the field in the parent that marks the beginning of the multiply-occurring fields.	All characters and digits, except special characters.

Field Attributes

Field Declarations:

```
FIELD[NAME]= {fieldname}          {alias          }
              {FILLER   } ALIAS= {rectype identifier},USAGE=format,
              {RECTYPE  }          {ORDER        }
              {MAPVALUE }          {KEY(n)       }
              {MAPFIELD }          {MAPFIELD     }
```

```
ACTUAL=format [,FIELDTYPE =I,ACCEPT{list }]
              [,INDEX=I           {range}]
              [,TITLE='text '][,DESCRIPTION=description],$
```

```
GROUP= {groupname|keyname} ,ALIAS={DKEY(n)|KEY(n)}
        ,USAGE=format,ACTUAL=format [,FIELDTYPE=I],$
```

```
[DEFINE name/format=expression;$]
```

Attribute	Function	Value
FIELDNAME	Uniquely identifies a data item in the file.	All characters and digits, with a maximum of 64.
ALIAS	Alternate field name, sometimes used to provide additional information about the field.	All characters and digits, except special characters, with a maximum of 64.
USAGE	Describes the format of the field as interpreted for usage (display) purposes.	See <i>RMS Attribute Summary</i> on page 32-42.
ACCEPT	Assigns a list or range of acceptable values for RECTYPE or MAPVALUE.	See <i>RMS Attribute Summary</i> on page 32-42.
ACTUAL	Describes the format of the field as it exists in the external file.	See <i>RMS Attribute Summary</i> on page 32-42.
FIELDTYPE	Identifies a group field on keyfield.	Can be only one character, which is I.
TITLE	Provides one or more lines of text to be used as an alternate column heading for the field name on reports.	All characters and digits with a maximum of 64.

Attribute	Function	Value
DESCRIPTION	Documents the meaning of a field in the Master File.	All characters and digits, with a maximum of 44.
GROUP	Identifies a key.	All characters and digits, except special characters, with a maximum of 48.
DEFINE	Creates a temporary field for reporting purposes.	Mathematical or logical statement made up of constants, field names, or other DEFINE fields.

RMS Attribute Summary

This topic contains a summary of the attributes.

Throughout the summary we refer to special characters and denote this reference with an “*”. You should avoid using the special characters listed below, as they may impose operational restrictions in some environments.

+ (plus sign)	' (apostrophe)	“ (double quotation marks)
- (hyphen)	; (semicolon)	< (less than sign)
\$ (dollar sign)	b/ (blank)	> (greater than sign)
* (asterisk)	, (comma)	= (equal sign)
/ (slash)	(concatenation symbol)	. (period)
() (parentheses)		

Attribute:	ACCEPT
Length:	1 to 255 characters
Example:	<code>ACCEPT = A OR B OR C</code>
Function:	<p>Is optional, and enables you to assign a list or range of acceptable values to a RECTYPE or MAPVALUE field in a file.</p> <p>The syntax is</p> <pre>ACCEPT = {list range}</pre> <p>where:</p> <p><i>list</i></p> <p>Is a string of acceptable values: value1 OR value2 OR value3...</p> <p>For example, ACCEPT=RED OR WHITE OR BLUE. You can also use a blank as an item separator. If the list of acceptable values runs longer than one line, continue it on the next line. The list is terminated by a comma (,).</p> <p><i>range</i></p> <p>Gives the range of acceptable values: value1 TO value2</p> <p>For example:</p> <pre>ACCEPT = 150 TO 1000</pre>

Attribute:	ACTUAL	
Length:	1 to 8 characters.	
Function:	Describes the type and length of a field as it actually exists in a file. The source of this information is an existing description of the file. The following chart shows the data format types that the server reads:	
	Type	Meaning
	An	Alphanumeric characters A–Z, 0–9, and other ASCII display characters, where $n = 1–256$.
	D8	8-byte double-precision floating-point numbers.
	F4	4-byte single-precision floating-point numbers.
	In	Binary integers: I1 Single-byte binary integer I2 2-byte binary integer I4 4-byte binary integer I8 8-byte binary integer
Pn	Packed decimal internal format. The length is the number of bytes, each of which contains two digits, except for the last byte which contains a digit and the sign (+ or -). P6 means 11 digits plus a sign, packed two digits to the byte, for a total of six bytes of storage, where $n = 1–8$.	

Function: (continued)	Type	Meaning																
	<i>Zn</i>	<p>Zoned decimal format (numeric string) where <i>n</i> is the number of digits (1 to 31), each of which takes one byte of storage. The last byte contains a digit and the sign.</p> <p>There are several standards for zoned data. For Read purposes, only right overpunched standards are supported and can be determined on Read since they are unique.</p> <p>The specific format to use when writing data must be known for read/write purposes. The default is ASCII right overpunch. To change the default, you must edit the EDACONF [.BIN]EDAENV.COM file and add the following logical:</p> <pre>DEFINE /NOLOG IBI_ZONED_OUT_TYPE {1 2}</pre> <p>where:</p> <p><u>1</u> Is the ASCII right overpunched standard (default).</p> <p><u>2</u> Is the EBCDIC right overpunched standard.</p> <p>Zoned right separate numeric or zoned left overpunched numeric formats are not supported.</p>																
	DATE	<p>Unless your file was created by a program, all of the characters will be in ASCII format type A (alphanumeric).</p> <p>The server permits the following conversions from ACTUAL format to USAGE (display) format:</p> <table border="1" data-bbox="523 1154 1303 1559"> <thead> <tr> <th data-bbox="523 1154 915 1207">USAGE</th> <th data-bbox="915 1154 1303 1207">ACTUAL</th> </tr> </thead> <tbody> <tr> <td data-bbox="523 1207 915 1261">I</td> <td data-bbox="915 1207 1303 1261">I</td> </tr> <tr> <td data-bbox="523 1261 915 1315">P</td> <td data-bbox="915 1261 1303 1315">P</td> </tr> <tr> <td data-bbox="523 1315 915 1369">P, D, I, F</td> <td data-bbox="915 1315 1303 1369">Z</td> </tr> <tr> <td data-bbox="523 1369 915 1422">D</td> <td data-bbox="915 1369 1303 1422">D</td> </tr> <tr> <td data-bbox="523 1422 915 1476">F</td> <td data-bbox="915 1422 1303 1476">F</td> </tr> <tr> <td data-bbox="523 1476 915 1530">P</td> <td data-bbox="915 1476 1303 1530">I8</td> </tr> <tr> <td data-bbox="523 1530 915 1559">YYMD</td> <td data-bbox="915 1530 1303 1559">DATE</td> </tr> </tbody> </table>	USAGE	ACTUAL	I	I	P	P	P, D, I, F	Z	D	D	F	F	P	I8	YYMD	DATE
USAGE	ACTUAL																	
I	I																	
P	P																	
P, D, I, F	Z																	
D	D																	
F	F																	
P	I8																	
YYMD	DATE																	

For a complete list of date formats, see *Segment Attributes* on page 32-3.

Note: The ACTUAL value of DATE indicates that the field is an OpenVMS 64-bit datetime stamp. This usage also requires an A4 filler field immediately following the date field in the Master File.

Attribute:	ALIAS
Alias:	SYNONYM
Length:	1 to 12 characters
Permitted Values:	All characters and digits.*
Example:	<code>ALIAS=CTY</code>
Function:	<p>Is an alternate name to identify a data field. Since references to data fields are based on the names, or aliases, or shortest unique truncation, it is useful to make the ALIAS a short abbreviation representative of the data. Short, simple names are best, with no embedded blank spaces or special characters.* For example, PART CODE or PART-CODE should be PARTCODE or PART_CODE, or PC.</p> <p>Names must begin with a letter, but can contain numbers. Do not use names that might conflict with report column names (C1, C2, ...), row names (R1, R2, ...) or HOLD file aliases (E01, E02, ...). Also, avoid reserved keywords such as BY or ACROSS.</p> <p>ALIAS has specialized functions when used with KEY definitions, RECTYPE and MAPFIELD, as discussed at the beginning of this manual.</p>

Attribute:	DEFINE
Example:	<code>DEFINE PROFIT/D8 = RETAIL_COST - DEALER_COST;\$</code>
Function:	<p>Is optional, and enables you to store definitions for temporary fields in your Master File for reporting purposes.</p> <p>The syntax is</p> <pre>DEFINE name[/format] = expression;\$</pre> <p>where:</p> <p><i>name</i></p> <p>Is a 1 to 48 character field name for the defined field.</p> <p><i>format</i></p> <p>Is the optional display format for the defined field, separated from <i>name</i> by a slash (/). The display format for the field follows the rules for USAGE formats described under field attributes. The default value is D12.2.</p> <p><i>expression</i></p> <p>Can be either a mathematical or logical statement, made up of constants, database fields, and temporary defined fields.</p> <p>The expression must end with a semicolon (;) followed by a dollar sign (\$). A DEFINE is placed at the end of the segment it is associated with.</p> <p>A defined field stored in a Master File can refer only to fields in its same segment. To create a defined field that refers to fields in other segments, create a temporary field using the DEFINE command prior to your report request.</p>

Attribute:	DESCRIPTION
Length:	1 to 44 characters
Permitted Values:	All characters and digits.
Example:	<code>DESCRIPTION=STANDARD COST CATEGORY,\$</code>
Function:	<p>Is optional, and is used only for documenting the meaning of a field in the Master File. It is ignored during processing. Since a Master File is a comma-delimited file that the server can read directly, it is possible to prepare reports in various formats whose subjects are the names and attributes of the data fields.</p> <p>If the DESCRIPTION contains a comma (,), the entire text must be enclosed in single quotation marks ('). For example:</p> <pre>DESCRIPTION='TOTAL COST, NOT NORMALIZED',\$</pre> <p>Another way to add comments to a Master File is to place them after the dollar sign (\$).</p> <p>Descriptions of DEFINE fields in the Master File are placed on separate lines.</p> <p>For example:</p> <pre>DEFINE ITEMS_SOLD/D8 = INVENTORY - ORDERED ;DESC=DAMAGED ITEMS NOT INCLUDED,\$</pre> <p>Note: When used on a DEFINE expression, the dollar sign (\$) does not immediately follow the semicolon (;).</p> <p>The semicolon (;) after a DEFINE must display on the same line as the attribute, which follows the DEFINE.</p>

Attribute:	FIELDNAME
Alias:	FIELD
Length:	1 to 64 characters
Permitted Values:	All characters and digits.*
Example:	<code>FIELD=INVENTORY</code>
Function:	<p>Identifies the data items in a file. Names must be unique within a file.*</p> <p>The full field name is used as the default title for the data printed on reports. Hence, names representative of the data should be selected.</p> <p>Names must begin with a letter, though they may contain numbers. Avoid names that might conflict with report column names (C1, C2, ...), row names (R1, R2, ...) or HOLD file aliases (E01, E02, ...). Also avoid reserved keywords such as PRINT, BY, ACROSS, and so on.</p> <p>FIELDNAME has specialized functions when used with RECTYPE and MAPVALUE, as discussed at the beginning of this manual.</p>

Attribute:	FIELDTYPE
Alias:	INDEX
Length:	1 character
Permitted Values:	I
Example:	<code>FIELDTYPE=I</code>
Function:	<p>Identifies a group field or a field that serves as a secondary key, which indicates that this secondary key can be used for direct retrieval. This attribute only applies to keyed (indexed) RMS files.</p>

Attribute:	FILENAME
Alias:	FILE
Length:	1 to 8 characters
Permitted Values:	All characters and digits, but no embedded blanks.*
Example:	FILENAME=EQUIP
Function:	Is optional, and is used for documentation purposes. It should correspond to the external file name of the Master File.

Attribute:	GROUP
Alias:	KEY
Length:	1 to 66 characters
Permitted Values:	All characters and digits, but no embedded blanks.*
Example:	GROUP=PUBKEY
Function:	<p>Is used to describe the primary key in a keyed file or a field that is composed of subfields. The USAGE format and ACTUAL format of the GROUP is calculated using the USAGE and ACTUAL formats of the FIELDS that comprise the GROUP.</p> <p>Names must begin with a letter, though they may contain numbers. Avoid using names that might conflict with report column names (C1, C2, ...), row names (R1, R2, ...), or HOLD file aliases (E01, E02, ...).</p>

Attribute:	PARENT
Alias:	PARENT
Length:	1 to 8 characters
Permitted Values:	All characters and digits, but no embedded blanks.
Example:	<code>PARENT=CARSEG</code>
Function:	<p>Is the name of the segment that is the parent or "owner" of the current segment. In the Master File, the information describing the parent must precede any reference to it as a parent.</p> <p>The first, or "root" segment, in a file cannot have a parent by definition, but the name "SYSTEM" may be used, or the attribute left blank. Every other segment must have a parent named. If no parent is named, the immediately preceding segment will be used by default.</p>

Attribute:	SEGNAME
Alias:	SEGMENT
Length:	1 to 8 characters
Permitted Values:	All characters and digits, but no embedded blanks.*
Example:	<code>SEGNAME=MODSEG</code>
Function:	Identifies a collection of data fields. Segments which have a relationship to each other must have unique names within a given file description.

Attribute:	SUFFIX	
Alias:	FILESUFFIX	
Length:	1 to 8 characters	
Permitted Values:	RMS	
Example:	SUFFIX=RMS	
Function:	In order to identify the type of file the description applies to, the SUFFIX is given one of the following values:	
	Value	Meaning
	RMS	A keyed RMS file, which uses an Access File to locate and access data using the RMS Index.
	ISAM	A keyed RMS file. ISAM is supported as an alternate keyword for backward compatibility to FOCUS 6.x applicaitons.

Note: SUFFIX values for other proprietary databases that can be accessed are described in the appropriate manuals for these options.

Attribute:	TITLE
Length:	1 to 64 characters.
Permitted Values:	All characters and values.
Example:	<code>TITLE= ' Products '</code>
Function:	<p>Is optional, and enables you to supply an alternate column title to replace the field name that is normally used.</p> <p>The syntax is</p> <pre>TITLE= ' text '</pre> <p>where:</p> <pre>text</pre> <p>Must be enclosed within single quotation marks (') and can contain 1 to 64 characters.</p> <pre>TITLE</pre> <p>cannot span more than one line in the Master File. If necessary, move the entire attribute to a line by itself. To display the TITLE on more than one line in the report, use commas to divide the text. To include blanks at the end of a column title, simply type them in and enter a slash (/) in the final blank position.</p> <p>TITLE attributes do not apply when direct operations (PCT., AVE., ...) are used on the field. To rename such columns, use the AS phrase.</p>
Changes:	<p>You can override both field names and TITLE attributes with AS phrases in your request. You can issue a SET TITLE command to change the default titles to either the field names or titles supplied in the Master File. You can change the TITLE attribute in the Master File.</p> <p>Note: Client tools using the API do not have access to the TITLE attribute. The TITLE attribute is available only with WebFOCUS.</p>

Attribute:	USAGE
Length:	1 to 8 characters
Function:	<p>Defines the report display format of the data field. This attribute defines the data field type, length, and any edit options that are to be applied when the field values are printed. Special date formats, which are actually an extended series of edit options, can also be used.</p> <p>The syntax is</p> <p><i>USAGE=format</i></p> <p>where:</p> <p><i>format</i></p> <p>Consists of three parts:</p> <p><i>tllleeeee</i></p> <p><i>t</i> = field type</p> <p><i>lll</i> = field display length</p> <p><i>eeee</i> = edit options</p>

Permissible USAGE field types and lengths are shown in the chart below:

USAGE	Length	Description
<i>A</i>	1–9,095	Alphanumeric text
<i>D</i>	1–19	Decimal, double-precision numbers
<i>F</i>	1–9	Decimal, single-precision numbers
<i>I</i>	1–11	Integer values (no decimal places)
<i>P</i>	1–17	Packed decimal numbers
<i>YYMD</i>	10	Displayed as YYMD

Edit options only affect printed or displayed fields. They are not active for extract files.

The following table summarizes the edit options and includes sample USAGE values:

Edit Option	Meaning	Effect
%	Percent sign	Percent sign Displays a percent sign along with numeric data. Does not calculate the percent.
B	Bracket negative	Encloses negative numbers in parentheses.
c	Comma suppress	Suppresses the display of commas. Used with numeric format options M and N (floating and non-floating dollar sign) and data format D (floating-point double-precision).
C	Comma edit	Inserts a comma after every third significant digit, or a period instead of a comma if continental decimal notation is in use.
DMY	Day-Month-Year	Displays alphanumeric or integer data as a date in the form day/month/year.
E	Scientific notation	Scientific notation Displays only significant digits.
L	Leading zeroes	Adds leading zeroes.
M	Floating \$ (for US code page)	Places a floating dollar sign \$ to the left of the highest significant digit. Note: The currency symbol displayed depends on the code page used.
MDY	Month-Day-Year	Displays alphanumeric or integer data as a date in the form month/day/year.
N	Fixed \$ (for US code page)	Places a dollar sign \$ to the left of the field. The symbol displays only on the first detail line of each page. Note: The currency symbol displayed depends on the code page used.
R	Credit (CR) negative	Places CR after negative numbers.
S	Zero suppress	Zero suppress If the data value is zero, prints a blank in its place.

Read/Write Usage Limitations of the Data Adapter for RMS

The data adapter has certain limitations when used for write purposes. The following topics discuss these issues and possible alternatives that may be used.

OCCURS Statements in a Master File

OCCURS statements in a Master File are not directly supported by the write portion of the data adapter. If an OCCURS statement is for a specific number of instances, an alternate Master File may be coded by using specific naming (for example, AMTDUE OCCURS=12 would be coded as AMTDUE01, AMTDUE02, and so on). The write application must then reference the alternate Master File using the specific fields.

For example:

```
SEGNAME=MTH, SEGTYPE=S0, PARENT=YEAR, OCCURS=12,$  
FIELD=PAYABLES, AP, I8 , I4 , $
```

would be coded as

```
SEGNAME=MTH, SEGTYPE=S0, PARENT=YEAR, $  
FIELD=PAYABLES01, AP01, I8 , I4 , $  
FIELD=PAYABLES02, AP02, I8 , I4 , $  
FIELD=PAYABLES03, AP03, I8 , I4 , $  
FIELD=PAYABLES04, AP04, I8 , I4 , $  
FIELD=PAYABLES05, AP05, I8 , I4 , $  
FIELD=PAYABLES06, AP06, I8 , I4 , $  
FIELD=PAYABLES07, AP07, I8 , I4 , $  
FIELD=PAYABLES08, AP08, I8 , I4 , $  
FIELD=PAYABLES09, AP09, I8 , I4 , $  
FIELD=PAYABLES10, AP10, I8 , I4 , $  
FIELD=PAYABLES11, AP11, I8 , I4 , $  
FIELD=PAYABLES12, AP12, I8 , I4 , $
```

Commit Processing

The Data Adapter for RMS, in conjunction with RMS itself, treats a single SQL statement as a complete unit of work. This means that the server automatically commits after every SQL statement. If the front-end application sends an SQL COMMIT or SQL ROLLBACK statement(s), it will be ignored by the data adapter.

SQL Commands

The following are acceptable SQL commands:

Command	Function
<code>SELECT</code>	Retrieves data for the entire table (*) or for specified columns.
<code>DELETE</code>	Removes one or more records from an RMS keyed file.
<code>INSERT INTO</code>	Adds data to an RMS keyed file.
<code>UPDATE</code>	Updates values of one or more columns in a record of an RMS keyed file.

SQL, MODIFY and MAINTAIN Operations

Releases prior to 5.2 of the server did not support multi-record SQL INSERT, DELETE and UPDATE except when done as a PREPARE plus MODIFY and MAINTAIN were not supported. This limitation is removed as of the 5.2 release. Recoding existing applications that used PREPARE is not required, all methods are supported.

CHAPTER 33

Getting Started in SAP Business Intelligence Warehouse (BW)

Topics:

- Preparing the SAP BW Environment
- Configuring the Query Adapter for SAP BW
- Extracting Data With SAP BW Queries
- Managing SAP BW Metadata
- Overview of SAP BW Reporting Concepts
- Producing SAP BW Requests

The Query Adapter for SAP BW allows WebFOCUS for SAP Business Intelligence Warehouse (BW) and other applications to access SAP BW data sources through the SAP BW Multi-dimensional (OLAP) model. The adapter converts data or application requests into native SAP BW statements and returns optimized answers sets to the requesting program.

For sample requests and output, see Producing SAP BW Requests.

Preparing the SAP BW Environment

SAP BW remote communications require:

- One of the following operating systems: Windows NT/2000, Sun Solaris, HP-UX, or AIX. Availability may depend on operating system release level.
- SAP BW BASIS 4.6 or higher and SAP BW 2.0B or higher.
- Installation of the SAP BW RFC/SDK on the server system. For details, see your SAP documentation.
- A TCP/IP connection to the SAP BW destination machine.

The following SAP Authorization profiles are required at a minimum. This list may change depending on your SAP BW Release and/or site specific Authorization Profiles. These authorization requirements are determined based on a 3.0A system.

LIST OF AUTHORIZATION OBJECTS, FIELDS AND REQUIRED VALUES

S_RFC: Authorization check for RFC access

Field	Value
ACTVT (Activity)	16 (execute)
RFC_NAME (Name of RFC to be protected)	RSAB, RSOB, SYST
RFS_TYPE (Type of RFC object to be protected)	FUGR (function group)

S_RS_ADMWB: Administrator Workbench - Objects

Field	Value
ACTVT (Activity)	03 (Display), 16 (Execute)
RSADMWBOB (Administrator Workbench object)	INFOBJECT

S_RS_COMP: Business Explorer - Components

Field	Value
ACTVT (Activity)	03 (Display), 16 (Execute)
RSINFOAREA (InfoArea)	* (All or specific InfoAreas)
RSINFOCUBE (InfoCube)	* (All or specific InfoCubes)
RSZCOMPID (ID of a reporting component)	* (All)
RSZCOMPTP (Component type)	CKF, REP, RKF, STR, VAR

S_RS_COMP1: Business Explorer - Components: Enhancements

Field	Value
ACTVT (Activity)	03 (Display), 16 (Execute)
RSZCOMPID (Name (ID) of a reporting component)	* (All)
RSZCOMPTP (Type of a reporting component)	CKF, REP, RKF, STR, VAR
RSZOWNER (Owner (Person Responsible) for a Reporting Component)	* (All)

S_RS_ICUBE: Administrator Workbench - InfoCube

Field	Value
ACTVT (Activity)	03 (Display))
RSICUBE OBJ (InfoCube Subobject)	DATA
RSINFOAREA (InfoArea)	* (All) or specific InfoAreas
RSINFOCUBE (InfoCube)	* (All) or specific InfoCubes

S_RS_ISET: Administrator Workbench - InfoSet

Field	Value
ACTVT (Activity)	03 (Display))
RSINFOAREA (InfoArea)	* (All) or specific InfoAreas
RSINFOSET (InfoSet)	* (All) or specific InfoSets
RSISETOBJ (InfoSet-Subobject)	DATA

S_RS_ODSO: Administrator Workbench - ODS Object

Field	Value
ACTVT (Activity)	03 (Display))
RSINFOAREA (InfoArea)	* (All) or specific InfoAreas
RSODSOBJ (ODS Object)	* (All) or specific ODS Objects
RSODSPART (Subobject for ODS Object)	DATA

Accessing Multiple Systems

The query adapter can operate across multiple R/3 systems. For each system, the query adapter requires one BW logon consisting of client, user, and password. This logon must be:

- RFC enabled.
- Authorized for all clients that you may access.

You may need additional authorizations, depending on the SAP BW data you wish to access. For details, see your SAP BW administrator.

Configuring the Query Adapter for SAP BW

Configuring the query adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to SAP BW, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Manually add the command in the global server profile (edasprof.prf) or in a user profile (user.prf).
- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the global server profile.

You can declare connections to more than one SAP R/3 data source by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to SAP R/3 takes place when the first query that references the connection is issued. If more than one SET CONNECTION_ATTRIBUTES command contains the same system, the query adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Syntax

How to Declare Connection Attributes Manually

Explicit. The user IDs and passwords are specified for each connection and passed to SAP BW for authentication and request execution.

```
ENGINE [BWBAPI] SET CONNECTION_ATTRIBUTES
system/user_ID,password:'client'
```

Password Passthru. The user ID and password received from the client application are passed to SAP BW for authentication and request execution.

```
ENGINE [BWBAPI] SET CONNECTION_ATTRIBUTES system/:'client'
```

where:

BWBAPI

Indicates the Query Adapter for SAP BW. You can omit this value if you previously issued the SET SQLENGINE command.

system

Is the SAP BW System ID. It must point to a valid system entry in etc/sapserv.cfg.

user_ID

Is the SAP BW user ID.

password

Is the password for the user logon.

Note: To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

client

Is the SAP BW client for the user logon. This value must be enclosed in single quotation marks (').

Procedure How to Declare SAP BW Connection Attributes From the Web Console

To declare SAP BW connection attributes from the Web Console:

1. Launch the Web Console.
2. Click *Adapters* on the left pane to open the Adapter Configuration Window.
3. Click *Add*, then *SAP BW* and then *SAP BW* to open the Add BW System Connect Parameters page in the right pane.

The configuration page prompts you for the following connection attributes. The settings are stored in `etc/sapserv.cfg` and the information is stored by blocks. A block is identified by the system ID (usually the three character name of the SAP BW instance).

User Authentication Parameters

The major work of the query adapter is to translate the user request into code that can be understood by SAP BW. For this purpose it requires a SAP BW user ID with a given set of privileges. In the following table this user ID is referred to as `IBI_USER`. After the request has been generated, and is ready for execution, it can be executed either under the `IBI_USER` ID, or under a less privileged user, referred to as `USER`; results are then based on the user's credentials.

Security	There are two methods by which a user can be authenticated when connecting to an SAP BW instance: Explicit. The user IDs and password are explicitly specified for each connection and passed to SAP BW for authentication and request execution. Password Passthru. The user ID and password received from the client application are passed to SAP BW for authentication and request execution.
System	Is the three character long alphanumeric ID that is used to describe the SAP connection. Using the instance name of the SAP installation is a common practice but not mandatory.

IBI_CLIENT	The SAP BW Client for the IBI logon, maximum three characters.
CLIENT	The SAP BW Client for the user logon, maximum three characters.
IBI_USER	The SAP BW user ID for the IBI logon.
USER	The SAP BW user ID for the user logon.
IBI_PASSWORD	The SAP BW password for the IBI logon, maximum eight characters.
PASSWORD	The SAP BW password for the user logon, maximum eight characters.

Server Internal Parameters

FPOOL	This parameter does not apply for the BW adapter unless the BW system is converted to SAP R/3. Leave the default value for BW.
DATACLASS	This parameter does not apply. Leave the default value.
TMPDIR	This parameter does not apply. Leave the default value.
RANGEBEG	This parameter does not apply. Leave the default value.
RANGEEND	This parameter does not apply. Leave the default value.

Connection Parameters

APPGRP	This parameter does not apply for the BW adapter unless the BW system is converted to SAP R/3. Leave the default value for BW.
MSGHOST	This parameter does not apply. Leave the default value.
HOST	The hostname of the SAP BW application server.
GWHOST	The hostname of the machine where the SAP BW gateway process is running. In the case where there is only one SAP BW application server, gwhost and host will be the same.
SYSNR	The SAP BW system number that is two characters.
LANGUAGE	The language installed on SAP BW to be used with the Query Adapter for SAP BW. It is one character long and defined in SAP system. (e.g. E for English, D for German).

Parameters for Mixed Character Code Set

eda2sap	This parameter does not apply. Leave the default value.
sap2eda	This parameter does not apply. Leave the default value.

Once all necessary parameter values are entered, click *Configure* to save the entry in the edasprof.prf file.

The newly added connection will be placed under the system name you entered within the SAP BW folder beneath the Configured folder in the left pane. The right pane with the connection parameters will be cleared, and it will only show the title Configuring Data Adapters.

Note: After configuring the adapter, it is necessary to install the static function modules into the SAP BW data dictionary to complete the configuration.

Extracting Data With SAP BW Queries

Queries are the methods for extracting data from the InfoCube. A query must be created or must pre-exist for reporting to be performed. Queries form query cubes that are used in creating report data.

Imagine the InfoCube as a multi-dimensional cube. A subcube (query) is cut from the InfoCube by the selection of characteristics and key figures. In this way, the data of the InfoCube can be quickly targeted and evaluated. The more precisely the query is defined, the smaller the subcube and the quicker the query can be navigated and refreshed. The query evaluates the entire data set of the InfoCube. Selecting certain characteristics means that they can be more closely analyzed while others remain unspecified. The resulting key figures are aggregated across all characteristic values for the unspecified characteristics.

A default navigational state is also established in the query definition when you arrange the characteristics and key figures in the rows and columns of the query.

A query is defined in SAP BW using the Excel-based BEx Analyzer. Once a query has been defined and released, the OLAP processor requests the data from the query and presents the current view of the stored data.

Only the data that is currently requested is transferred from the InfoCube to the query. The OLAP processor builds the query from the InfoCube data and provides methods for navigating through the data in several dimensions. Since a query pre-selects information, the same InfoCube can yield dramatically different results depending on the query used to view its contents.

Defining New Business Explorer Queries

You must define a SAP Business Explorer query before reporting from SAP BW InfoCube data. This query serves as a template for data extraction from the cube.

Note: The information that follows is based on SAP BW Business Explorer documentation. SAP BW BEx documentation is available from <http://help.sap.com>.

Procedure How to Select an InfoCube to Query

1. The Business Information Warehouse must contain at least one InfoCube before you can define a new query. Start the Business Explorer Analyzer.
2. From the BEx toolbar, choose *Open*. You will see the selection screen for all existing workbooks.
3. Choose *Queries*. The selection screen displays all available queries.
4. Choose *New*. You will see the selection screen for all InfoCubes for which you can define a new query or queries.
5. Select the InfoCube that has the data on which the query should be based. To display technical names for InfoCubes, set the Technical name on/off icon to *On*.

Tip: Since InfoCubes can have similar names for different elements, when building the query we recommend that you use technical names, rather than friendly names, as a reference. For related information, see *Understanding Field Names* on page 33-13.

The available objects in the InfoCube you have selected are displayed as a directory tree in the left part of the screen.

Next, you will select the objects for the query and drag them to the appropriate boxes to build the query.

Procedure How to Create a Query

The right area of the BEx screen contains selection boxes for the filter selection, the rows, the columns, and the free characteristics of the query.

Perform the following steps to create a query:

1. Click the plus or minus sign to the left of the dimension or Key Figures you want to query.
The object list will expand and display a list of all the available key figures or characteristics .
2. Drag and drop characteristics and key figures from the InfoCube into the selection box of the query definition.

These may be filters, rows, columns, and free characteristics.

Procedure How to Filter a Query

You can filter queries in order to place restrictions on them. Filter selection restricts the entire query, meaning that all of the InfoCube data is aggregated with the filter selection. To select fields you want to use to filter the query, complete the following:

1. From the object list of the InfoCube, select the characteristics or the key figure upon which the query should be based.

Note: Since they are used in definitions, fields selected as filters are not be displayed in the Query Adapter for SAP BW metadata. They are used to screen the data and thus contain no information to be reported on. If you wish to screen data and report from it, see *Restricting Query Characteristics* on page 33-10 or *Restricting and Calculating Key Figures* on page 33-11.

2. Drag the object to the Filter box.
3. Right-click the object in the filter box. A dialog box opens showing the possible filter definitions for the object.
4. Select either a single member, a range of members, or a variable for the filter.

Restricting Query Characteristics

When defining a query, you may restrict characteristics to a single characteristic value, a value interval, a hierarchy node, or a characteristic value variable.

Procedure How to Restrict Characteristics

1. Choose the characteristic from the InfoCube for which you want to select a value range.
2. Drag the characteristic into the appropriate selection box of the query definition (rows or free characteristics).
3. Select the characteristic you wish to restrict (or filter). Using the right mouse button, select *Restrict* from the Context menu.
4. Choose whether you want to restrict the characteristic to a single value, a value interval, or a hierarchy node.

Tip: You can enter the characteristic values or hierarchy nodes you want to use, or you can display a list of all possible values by clicking the magnifying glass to the right of the input field.

5. Confirm your entries by clicking *OK*.

Restricting and Calculating Key Figures

You can restrict key figures to characteristic values, characteristic value intervals, or hierarchy nodes. For example, a restricted key figure would be Sales revenue in 1st quarter.

You can also restrict the key figures of the InfoCube for the query definition, or, using a formula, you can calculate new key figures from the (basic) key figures:

- **Restricted key figures.** (Basic) key figures for the InfoCube that are restricted (filtered) by selecting one or more characteristics.
- **Calculated key figures.** Formulas that consist of (basic) key figures for the InfoCube and/or calculated key figures that have already been created.

Procedure How to Restrict Key Figures

1. Drag a (basic) key figure into the key figure selection box. Alternatively, select the header of the selection box for rows or columns and, using the right mouse button, select *New Structure* from the Context menu.
2. Select the Structure directory, and, using the right mouse button, choose *New Selection* from the Context menu. The New Selection screen opens.
3. Enter a description of the restricted key figure in the text fields located in the upper part of the screen.
4. Underneath the text fields, on the left, is the directory of all the objects available in the InfoCube. Use the empty field on the right-hand side of the screen for the definition of the new selection.
5. Using drag and drop, choose a key figure from the InfoCube, and restrict it using a selection of one or more characteristic values.
6. Select *OK*. The newly restricted key figure is defined in the structure.

Procedure How to Calculate Key Figures

1. Create a new structure in the rows or columns of the query definition by highlighting the row or column directory using the right mouse button and selecting *New Structure* in the Context menu.
2. Drag a (basic) key figure of the InfoCube into the directory of the new structure.
3. Select the Structure directory, and choose *New Formula*. The Formula Definition screen opens.
4. Enter a description of the formula in the text fields located in the upper part of the screen.

Note: The entry field for the formula is underneath the text fields. In the bottom left of the screen are all of the operands available for the formula definition. These are the key figures that you have already defined in the structure, and all of the formula variables in the Variables directory that have been created in the variable maintenance.

The functions available as operators are on the right-hand side of the screen. These are symbols for the basic arithmetic operations and directories with calculation functions such as percentage or trigonometric functions. To the right of the operators is a number block.

Procedure How to Define a Formula

1. Choose the operands you want to use, and insert them in the entry field for the formula by double-clicking or by using drag and drop.
2. Choose the calculation functions you want to use by either clicking the symbols for the basic arithmetic operations, double-clicking to select the individual values or drag the entire key figure into the formula box.
3. Select the number values for the formula by clicking the number block.
4. Define your formula using the available operands and operators.
If you want to use a variable that is not contained in the operands, you must create the variable first.
5. Check the formula definition for correctness by pressing the scale icon.
6. Enter the name of the formula column in the description box.
7. Select *OK*. The newly calculated key figure is defined in the structure.

Viewing Query Properties and Releasing for OLAP

- To view the properties of a query, click the Query Properties icon on the toolbar. The Query Properties dialog box opens.
- To release a query for OLAP, click the Query Properties icon on the toolbar and check *Release for OLE DB for OLAP* in the Query Properties dialog box.

This enables the query to be displayed as a QUERY_CUBE for reporting purposes. The query elements (hierarchy levels, measures, variable, and properties) will be mapped to corresponding OLAP elements to create a synonym.

Managing SAP BW Metadata

SAP BW field names are generated from the Level Caption loaded in BW Master Data and obtained from the BW Query Cube. These captions provide friendly (person-readable) field names. However, the Query Adapter for SAP BW enables you to use either friendly or technical (invariant) field names when generating metadata. Technical field names are most useful for stable queries, while friendly field names are most useful for queries that may change frequently.

Note: You must generate an SAP BEx Query, release it for OLAP, and run the query *before* you can create a synonym. You will complete those tasks using the SAP Business Explorer, a tool that creates the SAP BW queries that iWay will report on. If you have not completed those tasks, see *Extracting Data With SAP BW Queries* on page 33-8, then return to Generating Metadata.

Understanding Field Names

The Query Adapter for SAP BW enables you to use either friendly or technical (invariant) field names when generating metadata. Use the technical field name approach when you are developing stable queries that are not often changed. Use the friendly field name approach when you are developing a rapid ad-hoc query, or when your query needs to be changed often.

Friendly names are generated from metadata element captions. Technical names are generated from the UNIQUE_NAME of the respective metadata elements: for example, Levels and Measures. The UNIQUE_NAME always contains the name of an object that is unambiguous within the context of the cube. Technical names are always SAP Language independent, and are less changeable across systems and releases. Since friendly field names can change when you migrate between SAP BW release levels or when you move reports between systems, technical field names are recommended in these circumstances.

For both friendly and technical field names, the names are uppercase and special characters (for example, spaces and commas) are automatically replaced with underscores. Suffixes are sometimes added to provide field name uniqueness through the Master File.

Example Friendly and Technical Field Names

Friendly Field Name

```
"Sales Volume"Title "Sales Volume"
```

Technical Field Name

```
"3IPY96CDK740I7444ILK15U1V" Title "Sales Volume"
```

Procedure How to Set Synonym Properties

Once you have created the queries, and before you create synonyms, you must set synonym properties.

1. Launch the Web Console.
2. Click *Configure*, then click *Edit Files*.
3. Open and edit the Edasprof.prf server profile.

At the end of the text in the file, add the following line:

```
ENGINE BWBAPI SET FIELDNAME {ALL|MEASURES|LEVELS|PROPERTIES|VARIABLES}  
{TECHNICAL|FRIENDLY}
```

You may select any combination of Measures, Levels, Properties, and Variables or choose ALL. Then, for the option(s) you have selected, you may choose to generate either a technical or a friendly field name. Your last combination of selections overrides any previous combination.

4. Navigate to the adapter configuration page. You will see all validly specified SAP BW connections.

Creating Synonyms

Synonyms define unique names (or aliases) for each SAP BW table or view that is accessible from a server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

Procedure How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click Metadata. The Editing Metadata page opens.

To create a synonym, you must have configured the adapter.

2. Expand the Add folder, expand the adapter folder, and then click a connection. The right pane displays selection options.

To display all available queries, leave the entry fields blank.

To filter by Catalog or Cube name, enter the name of the Catalog (InfoCube) or Cube (Query Cube) in the appropriate box. You can use the wildcard character (*) as needed. For example,

OCCA*

SYSTEM B2I

Filter by:

Catalog name Sample: ZPAY003

Cube name Sample: ZPAY003/ZPAYODS1

Select Synonyms

For a query cube, enter the full catalog name, followed by a forward slash (/), followed by the query cube name. You can use the wildcard character (*) as needed. For example:

OCCA_C01/ZALL*

3. Click *Select Synonyms*. All queries that meet the specified criteria are displayed.

Select Application Directory: baseapp ▾

All Select Create Synonym

Check	Default Synonym Name	Catalog Name	Cube Name
<input type="checkbox"/>	IDES_CCA_C01_Q000E	DCCA_C01	DCCA_C01/IDES_CCA_C01_Q000E
<input type="checkbox"/>	ZALLCAM1	DCCA_C01	DCCA_C01/ZALLCAM1
<input type="checkbox"/>	ZALLQUERY	DCCA_C01	DCCA_C01/ZALLQUERY
<input type="checkbox"/>	ZALLSTATE	DCCA_C01	DCCA_C01/ZALLSTATE
<input type="checkbox"/>	ZALLSTATE2	DCCA_C01	DCCA_C01/ZALLSTATE2
<input type="checkbox"/>	ZCAMT1	DCCA_C01	DCCA_C01/ZCAMT1
<input type="checkbox"/>	ZCCA	DCCA_C01	DCCA_C01/ZCCA
<input type="checkbox"/>	ZCCA2	DCCA_C01	DCCA_C01/ZCCA2
<input type="checkbox"/>	ZQ978	DCCA_C01	DCCA_C01/ZQ978
<input checked="" type="checkbox"/>	ZVRT	DCCA_C01	DCCA_C01/ZVRT

4. From the Select Application Directory drop-down list, select a directory. The default value is baseapp.
5. Complete your query selection:
To select all queries in the list, click *All*.
To select specific queries, click the corresponding check boxes.
6. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.
7. Click *Create Synonym*. Synonyms are created and added under the specified application directory.

Reference **Managing Synonyms**

In the navigation pane, click the name of the synonym to access the following options:

Properties	Displays a graphic representation of the synonym and enables you to edit its metadata.
Edit Master File	Enables you to manually edit the synonym's Master File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Master File, to update the synonym.
Edit Access File	Enables you to manually edit the synonym's Access File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Access File, to update the synonym.
Drop	Deletes the synonym.
Copy to	Copies the synonym to another application directory. Click the target directory from the resulting list.
Move to	Moves the synonym to another application directory. Click the target directory from the resulting list.

Mapping Metadata

OLAP (OLE DB standard) uses the following terms for row sets: Hierarchies, Levels, Measures, Members, and Dimension Properties. The corresponding terms are mapped in BW and the Query Adapter for SAP BW.

It is a good practice to regenerate the adapter's metadata whenever data is loaded into the cube. Regeneration keeps data up to date and enables the adapter to accurately record the extent to which each dimensions hierarchy expands.

Reference Mapping BW InfoCubes to OLE/DB Cubes

Business Information Warehouse (BW) InfoCubes are similar to OLE/DB cubes. Both are the central metadata objects and containers for data used for reporting. InfoCubes contain two types of data: key figures and characteristics. For related information, see *Extracting Data With SAP BW Queries* on page 33-8. For examples of these mappings, see *Mapping Illustrations* on page 33-19.

OLAP	BW	FOCUS
Cube	Query	Master File.
Dimension	Characteristic	Internal use appears as comment in Master File (see above).
Hierarchy	Hierarchy	Dimension (A Query Adapter for SAP BW Dimension is a group of fields connected by means of the WITHIN keyword. The WITHIN keyword relates inner values to outer values for summary purposes.
Level	Level	Field (the WITHIN keyword points to the parent field).
Member	Characteristic Value	Field values.
Dimension Property (including special property Key)	Attribute	Field (only printable when the reference Dimension Hierarchy is used as a BY field).
No OLAP mapping	SAP BW variable	Field (specially mapped in a Master File) that takes optional or required input.
Measure	Key Figure (Measure)	Numeric fields upon which FOCUS verbs (Print/Sum) can perform operations.

Example Mapping Illustrations

The following lines of syntax illustrate the makeup of the Query Adapter for SAP BW metadata. Each is listed with its corresponding definition:

```
FILENAME=WAREHOUS, SUFFIX=BWBAPI,$
```

This denotes the cube descriptor. The FILENAME=*value* is generated from the query cube name.

```
SEGNAME=WAREHOUS, SEGTYPE=S0,$
```

This is the segment name (used internally in FOCUS only). The SEGNAME=*value* is generated in the same way as FILENAME=*value*.

```
$ DIMENSION -MEASURES-
```

This is the beginning of the measures section of the Master File. The Fact Table is logically a dimension and is displayed accordingly in the Master File.

```
$ CARDINALITY=n
```

The cardinality of a dimension defines how many members can be retrieved for a report. If the Cardinality of a dimension is high, you should consider a WHERE test to restrict the returned values.

```
FIELD=STORE_INVOICE, ALIAS='STORE INVOICE', USAGE=D10.2, ACTUAL=D8.2, MISSING=OFF,$
```

This describes the measure. The first section in the Master File defines the fields that comprise the Fact Table for your cube.

```
$ DIMENSION: dimension_name
```

This is the Dimension section comment. The Query Adapter for SAP BW synonym logic puts dimension names in comments before descriptions of the corresponding hierarchies and increases the readability of the Master File. All columns that follow a dimension comment are members of the dimension.

Note: Multiple hierarchies for the same dimension display as separate dimension entries.

```
FIELD= PRODUCT_FAMILY, ALIAS='PRODUCT FAMILY', WITHIN= *PRODUCT, USAGE=A20, ACTUAL=A20, MISSING=OFF,$
```

This is the top level of a dimension hierarchy. Each subsequent level of the hierarchy will have a corresponding field name. Each level of the hierarchy has a following comment line with the level unique cardinality (count of members for the level).

Note: The WITHIN=*value* cannot exceed 66 characters. If the hierarchy unique name is longer than 65 characters (plus 1 character for *), Create Synonym generates a message. Although OLAP supports summary levels (the root level in the hierarchy), these are not displayed in the Master File.

```
FIELD=STORE_MANAGER, ALIAS='STORE MANAGER', USAGE=A20, ACTUAL=A20,  
MISSING=ON, §
```

This describes the Dimension property. Any fields listed in a dimension section of the Master File that do not have an assigned hierarchy (for example, no WITHIN), represent dimension properties. In this example, the property stores the store manager's name for the store in the rollup.

Reference Syntax Conventions for Master Files

The following conventions must be followed in a Master File that is used with SAP BW:

- **Missing=OFF is required.** There is no null data in a cube. Joins are not supported in convention with BW syntax. All fields in the Master File against a real cube should be explicitly set to MISSING=OFF to prevent any attempt to retrieve missing data, which would cause OLAP to return an error.
- **Quotes are required.** Any field or hierarchy name containing blanks must be enclosed in single quotation marks.
- **Hierarchy levels.** Hierarchy levels do not have clearly defined data types. As a result, the levels are always represented as alpha.
- **Property fields.** Can only be used if their associated dimension has been activated. For example, if you need to use BY <Property> you need to also have BY <dimension> NO PRINT in the same request. When you use "by dimension" in the request it will activate that dimension and all its properties. The NO PRINT option ensures that the field is not displayed in the resulting report. For related information, see *Reporting Rules* on page 33-23.
- **Special Property - Key.** BW dimensions have a default hierarchy noted by the name of the dimension appended with level_01. These default hierarchy fields have a special property called a Key that contains the BW internal reference code for the field. A Key may be printed for only the default hierarchy and may be used only when it is referenced. A Key field may also be used in WHERE or DEFINE statements, providing the dimension field is referenced. For related information, see *Reporting Rules* on page 33-23.

Example Dimension: OMATERIAL

The following example shows what can be expected when selecting the fields in a report. It is important to remember that two fields are returned for the same data value.

Field:OMATERIAL_LEVEL01

Field:OMATERIAL_KEY

as displayed in a report:

<u>OMATERIAL_LEVEL_01</u>	<u>OMATERIAL_KEY</u>	<u>TOTAL SALES</u>
Fitdrink 2000(CAN)	R100032	\$1000.00

Both fields represent the *same* characteristic value.

OMATERIAL_LEVEL_01 is the dimension level containing the captions for all members of the dimension.

OMATERIAL_KEY contains the SAP BW technical name for the members of the dimension.

Variable Types

SAP BW supports four types of variables:

- **Members.** Can be displayed in a separate TABLE request. When used in a TABLE query, a search criterion must be supplied if the variable type is mandatory (and there is no default value). The search criterion must render a set of variable values that correspond to the variable, which may be single value, interval, or complex (for complex there is no restriction on combinations). Each search predicate (an AND logical expression) can reference only one variable and no other cube elements.
- **Nodes.** Can be displayed in a separate TABLE request. Node type variables have a single value selection type.
- **Hierarchies.** Can be displayed in a separate TABLE request. Hierarchy type variables have a single value selection type.
- **Numeric.** Cannot be displayed.

For related information, see *Reporting With Variables* on page 33-25.

Overview of SAP BW Reporting Concepts

The SAP Business Information Warehouse (BW) is a data warehouse solution constructed with specialized content for reporting and analysis. Data is extracted from source systems, or InfoSources. After being scrubbed (that is, revised in programmatic ways for easier reporting), the data is loaded into InfoObjects (special reporting tables) that are combined in business-relevant ways to become InfoCubes (multi-dimensional analysis structures).

InfoCubes contain key figures (numeric value fields) and characteristics that describe the key figures (organization and classification information).

Sample key figures are:

- Value, which includes costs, sales, and sales deductions.
- Quantity, which includes number of employees, sales quantity, and amount of billings posted.

Sample characteristics are SAP BW organizational units, such as:

- Controlling area
- Company code
- Business area
- Division

Characteristics form dimensions that enable you to analyze the key figures from multiple perspectives.

Reporting Rules

Key Figures. Key figures that are pre-aggregated in SAP BW must use the SUM verb (most InfoCube data). Non aggregated Key Figures can use the detail verb PRINT. (See your data administrator for details.)

Properties. Properties must reference their parent dimension levels. (For related information, see *Syntax Conventions for Master Files* on page 33-20).

Variables. Variables must supply the appropriate syntax for the type (see *Reporting With Variables* on page 33-25).

Hierarchies. A single hierarchy from the same dimension can be represented on the report.

Cardinality. Because SAP BW is a multi-dimensional database, each field represents a multiplicative value for the total number of cells returned. (If the Cardinality in the Master File for the dimension is very large, you may want to use a WHERE test to limit the result.) For example:

```
TABLE FILE BW
SUM NET_SALES GROSS_SALES TOTAL_SALES (MEASURES CARDINALITY 3
BY COMPANY_CODE (DIMENSION 1 CARDINALITY 100)
BY REGION (DIMENSION 2 CARDINALITY 6)
END
```

is calculated as follows

(100 X 6 X 3 = 1,800 possible cells)

where:

Dimension 1 = 100 members X

Dimension 2 = 6 members X

3 Measures = 1,800 possible cells

Adding the following line of code

```
BY SALES_REP (DIMENSION 3 CARDINALITY 50)
```

changes the calculation to

(100 (D1) X 6 (D2) X 50 (D3) X 3 (measures) = 90,000 (possible cells)

General Tips for Reporting

When creating a report request:

- Do not use PRINT against Key fields inside OLAP cubes. Calculated and pre-aggregated fields need to be referenced by the equivalent of the verb that generated them. Reference count fields using the COUNT command, rather than PRINT or SUM. Reference SUM fields (aggregations) with the SUM command.

When reporting from ODS (Operational Data Store) or InfoSet Queries (ODS joins), depending on the summarization setting of the key fields, you may use the PRINT verb. When using PRINT against ODS objects, the BY fields (not to be confused with Key Figures Fields) determine the summarization level of unique values.

- To maximize reporting efficiency, always sort in the same direction as the hierarchy for the cube.

For example, if the hierarchy for a dimension is

Country > State_Province > City

sorting by City first, then State_Province results in a very inefficient request. Also, because of the hierarchical structure of OLAP, you retrieve only one BY field for each SUM or PRINT field in the report.

Using Columnar Reports to Slice Cubes

If you run a standard TABLE or FML report against a cube, you are running what is known as a Slice report. This two-dimensional report resembles a slice of a larger multi-dimensional cube. A Slice report flattens out the multi-dimensional structure of a cube to show information from one angle.

To create a valid Slice report, you must understand the cube's data structure and the logic and assumptions that the Query Adapter for SAP BW uses when slicing a cube.

Reference Valid and Invalid Slices-Cube Slicing Logic

The Query Adapter for SAP BW has built-in logic that enables it to determine the rows and cells to extract from rollups in order to deliver Slice reports. TABLE uses the lowest level BY phrase for each dimension to determine at which rollup to locate the lookup for aggregations. It then rolls the result up from there, performing aggregations on the data as needed to ensure the consistency of the results.

If you do not specify a BY field in your TABLE request, you may receive invalid data or a message. This happens because without a BY field to determine the level of rollup at which to perform the data lookup, TABLE does not know where to source the result.

Example Displaying Single Dimensions in a Slice Report

```
TABLE FILE MYCUBE
SUM STORE_COST STORE_SALES
BY COUNTRY_LEVEL_01 ON TABLE COLUMN_TOTAL
END
```

This report displays aggregate data sorted against one dimension in the source. The standard measures such as STORE_COST and STORE_SALES are referenced with SUM. Column totals are requested in the report.

Example Displaying Multiple Dimensions in a Slice Report

```
TABLE FILE MYCUBE
SUM STORE_COST STORE_SALES
PRINT PROFIT
BY COUNTRY_LEVEL_01 BY STATE_PROVINCE_LEVEL BY CITY_LEVEL_01
END
```

This report displays the same data as the previous one, except that it sorts against both dimension hierarchies in the cube. The order of the sort fields follows the logical order of the dimension hierarchy.

Reporting With Variables

A BW query uses variables to supply values at execution time. A variable can have an Entry_Type of either Mandatory/Required or Optional. All mandatory variable values must be supplied in the TABLE or SQL request. A WHERE or IF statement in the request provides the run time value(s). For example, if the variable COMPANY_CODE is mandatory, it will supply values for company code 430 at run time:

```
WHERE COMPANY_CODE_N EQ '430'
```

All variables passed to BW have a format of 124 alpha bytes. The synonym's Master File breaks variables into two parts: variable_C and variable_N.

- The caption (_C) is the first 60 bytes and is an internal reference for BW.
- The name (_N) is the last 64 bytes in which the actual key value will be sent to the query.

In the previous example, the name (_N) supplies the variable value. You can use either the caption or the name for screening, but you must supply appropriate values for each:

```
company_code_n : WHERE COMPANY_CODE_N EQ 430
company_code_c : WHERE COMPANY_CODE_N EQ 'IDES USA'
```

Tip: If the associated dimension of this variable is available in the query, do not build the selection on that field, but, rather, use the name. In other words, WHERE COMPANY_CODE_LEVEL_01 EQ 430 *will not* populate the variable, while COMPANY_CODE_N will. Additional non-variable selections are available, but they are less efficient. The non-variable conditions are applied after the BW answer set has been extracted.

Variables cannot be displayed in a request. They should be used only for applying selection criteria. You can display the dimensions or measures with which they are associated. Following are annotated examples of a valid request and an invalid request.

Valid request:

SUM ACCUMULATE_BALANCE	Measure ACCUMULATE_BALANCE (SUM verb must be used)
BY COMPANY_CODE_LEVEL_01	Dimension level COMPANY_CODE_LEVEL_01
BY COUNTRY__KEY__	Property COUNTRY__KEY__ belongs to COMPANY_CODE_LEVEL_01
WHERE COMPANY_CODE_N EQ '430'	Variable

Invalid request:

SUM ACCUMULATE_BALANCE	Measure
BY COMPANY_CODE_N	Variable (variable cannot be printed. COMPANY_CODE_LEVEL_01 should be used.)
BY COUNTRY__KEY__	Property (a property belongs to a dimension level and both must be used in the request.)
WHERE COMPANY_CODE_N EQ '430'	Variable

You cannot display a variable as a BY field. Dimension properties must be used in combination with their reference dimension. Variables can have a selection type of single, interval, or complex.

- Single type variables must have only one value supplied in the selection condition.
- Interval variables can have a single value or a range of values.
- Complex variables have no restrictions.

To supply a range, use two WHERE statements or combine them into one statement using AND. For example, this report needs to specify two years:

```
WHERE FISCAL_YEAR_N EQ "2001"  
      OR FISCAL_YEAR_N EQ "2002"  
WHERE FISCAL_YEAR_N GE "2001"  
WHERE FISCAL_YEAR_N LE "2002"  
WHERE FISCAL_YEAR_N GE "2001"  
      AND FISCAL_YEAR_N LE "2002"  
WHERE FISCAL_YEAR_N IN ( "2001", "2002" )  
WHERE FISCAL_YEAR_N FROM "2001" TO "2002"
```

Selections that use an OR connector, like the first sample above, can only test against the same field or related hierarchy. You cannot combine variable selection and non-variable selection in the same WHERE statement (screening predicate).

Working With Hierarchies

In SAP BW release 3.0A and lower, one characteristic of the query can be expanded in a hierarchical display.

Reference Creating Dynamic Hierarchies

The Query Adapter for SAP BW generates each level of a hierarchy with a field name corresponding to each level. For a dimension called Business Number and a hierarchy named States, level 1 will be States and level 2 will be the Business Number. You can create a report referencing all the fields of a hierarchy at once, or create it as an OLAP trigger report, referencing the top level of the hierarchy and enabling interactive drill down.

In SAP BW 3.0B and higher, multiple characteristics of the query can be expanded in a hierarchical display.

Creating Dynamic Dimensions

Using the Query Adapter for SAP BW, you can:

- Create dynamic dimensions by using the Create New Field (Define) function.
- Use hierarchy levels and dimensions that are not contained in the query cube, but are defined or calculated at run time.

Example **Creating a Dimension**

The following request creates a dimension called Daily Sales, which concatenates the company name, the current date (picked up from an the &DATE variable), and the Sales figures for that date.

```
DEFINE Daily Sales/A124= company|&date|'Sales'
END

TABLE FILE MYQUERY
SUM Retail_Amount
BY Daily Sales
END
```

The output would look as follows:

Daily Sales	Retail_Amount
ACME 11/15/2002	10,000
SMCE 11/15/2002	15,000

BW Explorer

WebFocus for SAP BW utilizes BAPIs to access to BW data. Under certain circumstances it might be suitable to use the R/3 adapter for BW, which actually creates dynamic ABAP/4 programs to read the data.

Procedure **How to Switch the BW system to an R/3 System**

To switch the BW system to an R/3 system:

1. Follow the steps defined in *Preparing The SAP R/3 Environment* of the *Data Adapter for SAP R/3* manual to create the Development Class or Package and the Function Group in the BW system and activate them.

Note: The SAP BW user ID should have the authorizations listed in the *Data Adapter for SAP R/3* manual in addition to the authorizations listed in this manual.

2. Follow the steps defined under the reference topic Declaring SAP BW Connection Attributes From the Web Console. Make sure to enter the function group name created on the previous step in the FPOOL field.
3. Once the Query Adapter for BW is configured, expand the Adapters tree on the left pane and navigate to the newly created SAP BW system under the Configured - SAP BW tree branch. Right-click the connection name and select *Properties* from the pop-up menu.

The connection parameters are shown in the left pane. Scroll down to see the *Switch BW System to SAP* checkbox next to the Configure button.

4. Select the checkbox and click *Configure*.

The SAP BW system has been moved under the Configured - SAP R/3 branch.

5. Click the system name.

The following menu items are shown:

Initialize BW Explorer	Creates synonyms for the BAPIs that BW Explorer uses.
Start BW Explorer	Starts the BW Explorer.
Test	This menu item runs a test procedure. The test procedure looks for an SAP R/3 table. The table does not exist in a BW system. Hence the output would be 0 records.
Properties	Displays a graphic representation of the synonym and enables you to edit its metadata.
Delete	Deletes the synonym.

6. Select *Properties* from the pop-up menu.
7. Scroll down and click *Install SAP Components* to install the necessary function modules into BW system.

For detailed information, refer to the *Data Adapter for SAP R/3* manual.

8. Click the system name to open the pop-up menu.
9. Select *Initialize BW Explorer*.

This might take a few minutes. Once finished, a message similar to the following will appear:

```

0  TRANSACTIONS:  0  TOTAL =  4  ACCEPTED=  4  REJECTED=  0
      SEGMENTS:           INPUT =  4  UPDATED =  0  DELETED  0
    
```

Using BW Explorer

Before using the BW Explorer, the connection for BW System should be switched to R/3. Refer to the steps defined under *How to Switch the BW system to an R/3 System* on page 33-28.

Procedure How to Use the BW Explorer

To use the BW Explorer:

1. Launch the Web Console.
2. Select *Data Adapters*.
3. Expand the SAP R/3 tree under the Adapters - Configured tree branch.
4. Click the connection name and select *Start BW Explorer* from the pop-up menu.

BW Explorer uses a set of BAPIs and displays the information as FOCUS reports.

- **List of Catalogs.** This report uses the BAPI_MDPROVIDER_GET_CATALOGS to list all the InfoProviders.
- **List of Cubes.** This report is a drill-down from the report List of Catalogs uses the BAPI_MDPROVIDER_GET_CUBES to list all the query cubes for the selected InfoProvider.

Drilling down from List of Cubes brings the report What do you want to Explore. Using this report, it is possible to drill-down to

- **Dimensions/Hierarchies/Levels/Members.** The first report shows the dimensions of the query. Further drill-down displays the Hierarchies for the selected dimension. Drill-down from the hierarchies shows the levels of the selected hierarchy and finally the drill-down from the hierarchy level displays the individual values for the selected hierarchy level.
- **Measures.** Displays information about the Measures (key figures) of the query.
- **Properties.** Shows the properties (attributes) associated with each dimension of the query.
- **Variables.** Shows the variables of the query, if any exist. If the report returns no records, it indicates that there are no variables assigned to the query.

Procedure How to Create Synonyms for BW Database Tables

Note: It is possible to read the database tables from BW using the Query Adapter for R/3. This method should be used only for master data reporting against very large volume of data, but the synonym would have to be created manually. The syntax is provided at the end of this section.

To create the synonym, the name of the BW database table should be determined beforehand. To determine the underlying database table for the InfoObject:

1. Logon to BW using the SAP front-end.
2. Launch the transaction RSA1. (Administrator Workbench: Modeling)

Make sure that InfoObject is selected on left pane. Find the InfoObject that you wish to access. To search, click the button with the binocular picture, and type the name of the InfoObject. Once the search result is displayed, double click on the name of the InfoObject to navigate in the main window.
3. On the main window, double-click the name of the InfoProvider.
4. Select the Master data/texts tab. You should see the master data table under *Master Data Tables*. For the syntax below, the table name /BI0/PMATERIAL is used as an example.
5. Start the server using edastart -t. The syntax for creating the synonym is as follows:

```
CREATE SYNONYM <app. directory>/<table name> FOR <table name> TABLE
DBMS SQLSAP AT <system name>
END
```

Example:

```
CREATE SYNONYM baseapp//BI0/PMATERIAL FOR /BI0/PMATERIAL TABLE
DBMS SQLSAP AT <system name>
END
```

Producing SAP BW Requests

The following requests and output are examples of the capabilities of the Query Adapter for SAP BW using standard ANSI SQL code. These examples require that CREATE SYNONYM has been performed on the relevant tables from the Web Console. You can use the Web Console to navigate the SAP application hierarchy by searching the relevant tables or by doing a simple search by name.

Example Retrieving Values From SAP BW

The following syntax retrieves values from an SAP BW database:

```
SELECT SUM(DISBURSEMENT),ADDRESS1_LEVEL_01 FROM ZPAY4  
GROUP BY ADDRESS1_LEVEL_01  
ORDER BY ADDRESS1_LEVEL_01;
```

The output is:

address1 Level 01	disbursement
1106 BROADWAY	132,242.33
11607 PACIFIC BLVD	9,174.96
11607 TREMONT AVENUE	14,849.56
1200 RIVERDALE	9,983.99
187 WEST 39TH ST	9,209.97
2601 8TH AVENUE	26,274.65
31-18 SOUTHERN BLVD	1,165.64
719 SEVENTH STREET	10,800.00
DRAWER 4, COOPER STA	21,406.32
P.O. BOX 1206	53,636.74
P.O. BOX 17616	6,703.90

Example **Selecting Items From a Hierarchy**

The following syntax selects items from a hierarchy in MARA:

```
SELECT SUM(SALES_VOLUME) , SUM(INCOMING_ORDERS) ,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_01 ,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_02 FROM ZBIGQ
GROUP BY PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_01 ,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_02
ORDER BY PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_01 ,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_02
```

The output is:

Product Hierarchy for material MARA Level 01	Product Hierarchy for material MARA Level 02	Sales Volume	Incoming Orders
Foods	Bakery product; configured to order	.00	400.00
Hardware	PCs	85,159,860.98	86,754,997.71
	Printer	.00	60,011.10
Lighting	Bulbs	62,235,898.00	62,674,320.00
Machines	Pumps	44,328,485.55	46,231,307.55
Paints	Gloss paints	.00	125.03
Services	Maintenance	291,562.54	485,315.83
Vehicles	Cars	155,640.00	155,640.00
	Motorcycles	41,761,619.76	42,494,811.00

Example **Selecting a Complete Hierarchy Path**

The following syntax retrieves a complete hierarchy path from MARA:

```
SELECT SUM(SALES_VOLUME) , SUM( INCOMING_ORDERS) ,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_01 ,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_02 ,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_03 ,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_04
FROM ZBIGQ
GROUP BY
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_01 ,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_02 ,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_03 ,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_04
ORDER BY
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_01 ,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_02 ,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_03 ,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_04 ;
```

The output is:

Product Hierarchy for material MARA Level 01	Product Hierarchy for material MARA Level 02	Product Hierarchy for material MARA Level 03	Product Hierarchy for material MARA Level 04	Sales Volume	Incoming Orders
Foods	Bakeryproduct; configured to order			.00	400.00
Hardware	PCs	Drive	Harddisk 1080 MB / SCSI-2-Fast	4,234,580.64	4,094,612.64
			Harddisk 2113 MB / ATA-2	5,216,171.71	5,216,171.71
			Harddisk 2149 MB / SCSI-2-Fast	4,727,005.36	4,713,724.40
			Harddisk 4294 MB / SCSI-2-Fast	5,132,812.80	5,132,812.80

Product Hierarchy for material MARA Level 01	Product Hierarchy for material MARA Level 02	Product Hierarchy for material MARA Level 03	Product Hierarchy for material MARA Level 04	Sales Volume	Incoming Orders
		Input device	Professional keyboard - MAXITEC Model	720,022.65	714,750.89
			Professional keyboard - NATURAL Model	813,765.24	788,838.81
			Professional keyboard - PROFITEC Model	576,079.36	576,079.36
			Standard Keyboard - EURO Model	620,294.21	600,911.21
			Standard Keyboard - EURO-Special Model	633,094.19	631,433.36
		Memory	SIM-Module 16M x 32, 70 ns	808,663.10	808,663.10
			SIM-Module 4M x 36, 70 ns	348,434.93	337,138.13
			SIM-Module 8M x 32, PS/2-72 Pin EDO-RAM	562,619.24	561,208.39
			SIM-Module 8M x 36, 70 ns	373,794.70	373,794.70

Product Hierarchy for material MARA Level 01	Product Hierarchy for material MARA Level 02	Product Hierarchy for material MARA Level 03	Product Hierarchy for material MARA Level 04	Sales Volume	Incoming Orders
		Monitor	Flatscreen LE 50 P	1,116,349.10	1,155,586.30
			Flatscreen LE 64P	1,749,563.60	1,749,563.60
			Flatscreen MS 1460 P	2,525,626.85	2,704,385.95
			Flatscreen MS 1575P	2,375,663.66	2,469,023.66
			Flatscreen MS 1585	3,365,928.24	3,487,309.32
			Flatscreen MS 1775P	3,809,366.74	4,074,466.82
			Flatscreen MS 1785P	4,722,113.10	4,722,113.10
			Jotachi SN4000	4,396,902.60	4,572,792.60
			Jotachi SN4500	2,473,317.57	2,647,587.95
			Jotachi SN5000	2,949,233.80	2,949,233.80
			MAG DX 15F/Fe	2,437,207.85	2,521,555.85
			MAG DX 17F	2,567,506.69	2,655,601.89
			MAG PA/DX 175	2,976,662.21	3,181,471.75
			PAQ MONITOR, 17", Color	683,666.40	685,463.40

Product Hierarchy for material MARA Level 01	Product Hierarchy for material MARA Level 02	Product Hierarchy for material MARA Level 03	Product Hierarchy for material MARA Level 04	Sales Volume	Incoming Orders
			PAQ Monitor, 20", Color	1,371,234.90	1,371,234.90
			SEC Multisync XV 17	3,557,594.77	3,690,080.79
			SEC Multisync XV15	2,961,072.00	2,961,072.00
			Sunny Extreme	3,215,185.42	3,339,353.42
			Sunny Tetral3	2,996,901.84	2,996,901.84
			Sunny Xa1	2,830,258.78	3,026,866.52
		PC ensemble	Maxitec-R3100 Personal Computer	3,616,021.08	3,600,521.08
			Maxitec-R 375 personal computer	194,520.00	194,520.00
		Processor	Motherboard 3100	.00	18,558.62
			Processor 100 MHz	401,608.95	401,608.95
			Processor 133 MHz	15,949.50	11,574.50
			Processor 166 MHz	1,083,067.20	1,016,409.60
	Printer	Laser printer	High Speed Printer	.00	60,011.10
Lighting	Bulbs	Light Bulb 40 Watt clear 220/235V		15,351,358.00	15,351,358.00

Product Hierarchy for material MARA Level 01	Product Hierarchy for material MARA Level 02	Product Hierarchy for material MARA Level 03	Product Hierarchy for material MARA Level 04	Sales Volume	Incoming Orders
		Light Bulb 40 Watt frosted 220/235V		5,432,920.00	5,256,060.00
		Light Bulb 40 Watt red 220/235V		2,976,965.00	3,197,425.00
		Light Bulb 40 Watt yellow 220/235V		3,032,064.00	2,917,008.00
		Light Bulb 60 Watt clear 220/235V		6,312,877.00	6,731,493.00
		Light Bulb 60 Watt frosted 220/235V		6,115,488.00	5,902,704.00
		Light Bulb 60 Watt red 220/235V		2,775,919.00	2,975,767.00
		Light Bulb 60 Watt yellow 220/235V		2,958,303.00	2,850,624.00
		Light Bulb 80 Watt clear 220/235V		7,095,740.00	7,589,438.00
		Light Bulb 80 Watt frosted 220/235V		5,120,425.00	4,935,200.00
		Light Bulb 80 Watt red 220/235V		2,416,380.00	2,416,380.00

Product Hierarchy for material MARA Level 01	Product Hierarchy for material MARA Level 02	Product Hierarchy for material MARA Level 03	Product Hierarchy for material MARA Level 04	Sales Volume	Incoming Orders
		Light Bulb 80 Watt yellow 220/235V		2,647,459.00	2,550,863.00
Machines	Pumps	Special pump	Pump GG IDESNORM 100-200	5,798,946.55	6,191,646.55
			Pump cast steel IDESNORM 150-200	6,721,540.00	6,962,580.00
			Pump cast steel IDESNORM 170-230	9,558,837.00	9,565,269.00
			Pump chrome-steel IDESNORM 150-200	475,602.00	495,702.00
			Pump standard IDESNORM 100-402	6,893,740.00	7,336,070.00
			pump CR IDESNORM 150-200 ATO	7,822,280.00	8,343,320.00
			pump sphere-cast IDESNORM 150-200	7,057,540.00	7,336,720.00
Paints	Gloss paints	Opaque	Coating Matt Green RAL 6014/10 Liter	.00	125.03

Product Hierarchy for material MARA Level 01	Product Hierarchy for material MARA Level 02	Product Hierarchy for material MARA Level 03	Product Hierarchy for material MARA Level 04	Sales Volume	Incoming Orders
Services	Maintenance	HiTech maintenance	PC Service (Configurable)	171,992.54	285,995.83
			PC Service Plus	119,570.00	199,320.00
Vehicles	Cars	Car (complete)	SAPSOTA FUN DRIVE 2000GT	155,640.00	155,640.00
	Motorcycles	Accessories	Motorcycle Helmet - Standard	3,651,535.18	3,800,706.24
		Components	Deluxe Gas Tank Striping Decals	658,897.52	703,373.20
			Deluxe Headlight	1,609,718.67	1,675,515.07
			Deluxe Taillight	546,588.47	579,120.40
		Motor-cycle (compl.)	CrossFun / 350 cm3	10,842,211.75	11,243,347.19
			IDES Glad Boy configurable	929,800.00	953,830.00
			SunFun / 1200 cm3	23,522,868.17	23,538,918.90

CHAPTER 34

Getting Started in SAP/R3

Topics:

- Preparing the SAP R/3 Environment
- Accessing Multiple SAP R/3 Systems
- Configuring the Data Adapter for SAP R/3
- Managing SAP R/3 Metadata
- SAP R/3 Table Support
- SAP R/3 Data Type Support
- SAP R/3 Open/SQL Support
- Advanced SAP R/3 Features
- Setting up the Report Processing Mode
- Supporting Mixed Code Page Environments
- Producing SAP R/3 Requests

The Data Adapter for SAP R/3 allows Data Migrator, ETL Manager, WebFOCUS for SAP, and other applications to access SAP R/3 data sources. The adapter converts data or application requests into native SAP R/3 statements and returns optimized answers sets to the requesting program.

For sample requests and output, see *Producing SAP R/3 Requests*.

Preparing the SAP R/3 Environment

You will use SAP GUI to logon to SAP R/3 and prepare the environment. You will need an SAP user ID and password to logon to SAP R/3.

The following SAP Authorization profiles are required at a minimum. This list may change depending on your SAP R/3 Release and/or site specific Authorization Profiles.

- User ID/password with Developer's Key, which must have been used at least once.
- S_TABU_DIS
- S_DEVELOP
- S_A.SCON (RFC)
- Ability to use the following SAP transactions:
 - S001: ABAP WORKBENCH
 - SE09: WORKBENCH ORGANIZER AUTHORITY
 - SE38: ABAP EDITOR
 - SE37: FUNCTION BUILDER

Note: If all the authorizations listed above are not available, the installation will fail due to SAP R/3 missing authorizations.

SAP R/3 adapter uses native SAP R/3 statements that are created dynamically during run-time. This is done via adapter component uploaded into SAP R/3 system during the configuration of the adapter. Prior to uploading the components, SAP R/3 system has to be prepared.

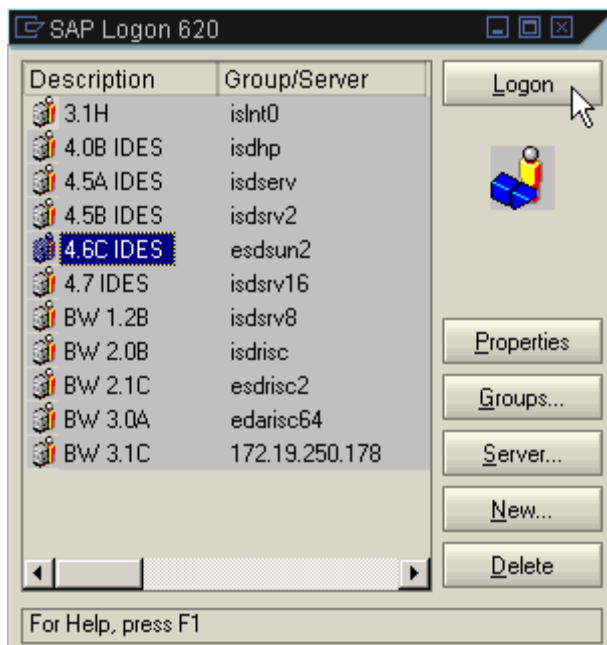
To prepare the SAP R/3 environment for the adapter components, perform the following steps:

1. Log on to SAP
2. Create Development Class or Package
3. Create Function Group
4. Deactivate Unicode checks - SAP R/3 4.7 overlay
5. Activate Function Group
6. Verify Function Group

Procedure How to Log on to SAP

1. Launch SAP logon. In a standard SAP GUI installation, the executable file resides in
\\Program Files\\SAP\\FrontEnd\\SAPgui\\saplogon.exe

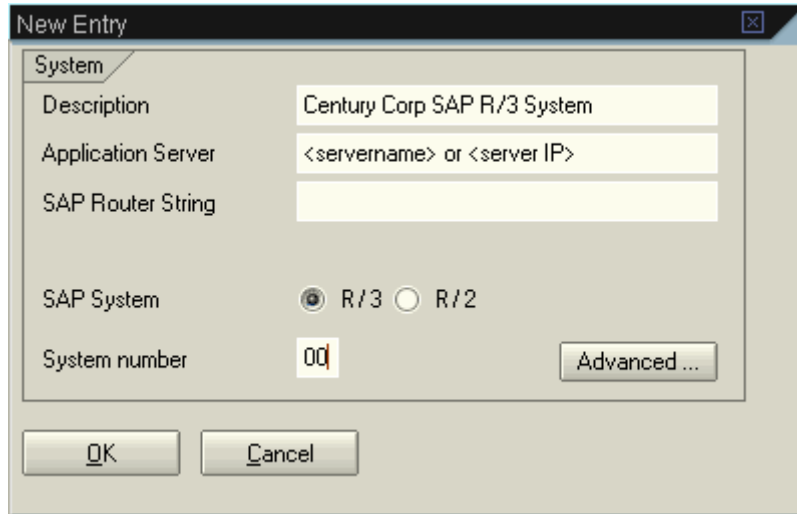
The SAP Logon dialog box opens:



If the list is empty, you create a new entry.

- a. Click *New...*

The New Entry dialog box opens.



- b. Fill the Description, Application Server and System Number. In most cases SAP Router String is not needed but check with your assigned SAP technical contact.

In the Application Server field, enter either the SAP application server's network name or IP address.

Click *OK* to create a new entry on the Logon list.

2. Select the entry on the list and click *Logon*.

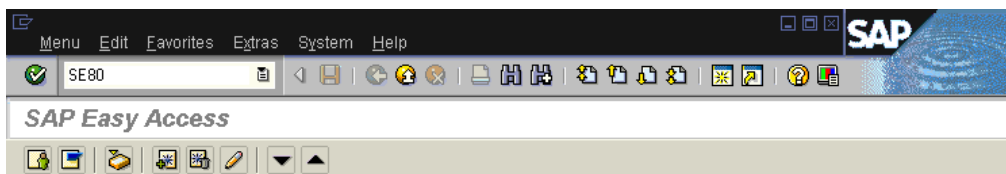
A logon page prompting for a user ID and password should come up. Logon to the system. Note that if the user ID is being used for the first time, the system will ask for the password to be changed.

Procedure How to Create Development Class or Package

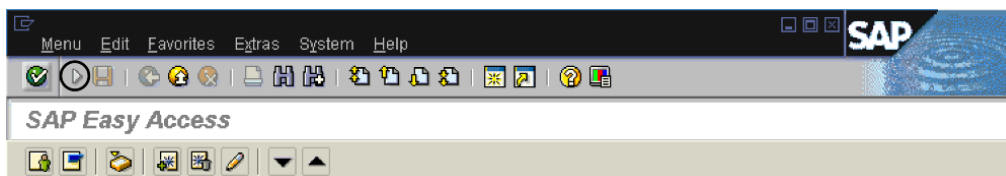
Note: SAP R/3 version 4.7 introduced Package as an alternative to Development Class. If you are using release 4.7, create a Package instead of a Development Class.

1. Execute transaction SE80 to start the Object Navigator.

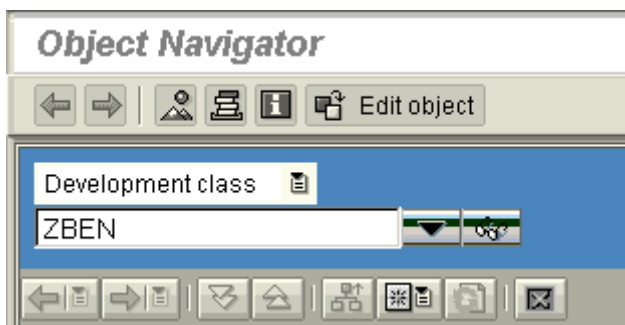
Hint: The easiest way to execute a transaction is to enter the transaction code on the command line box shown below.



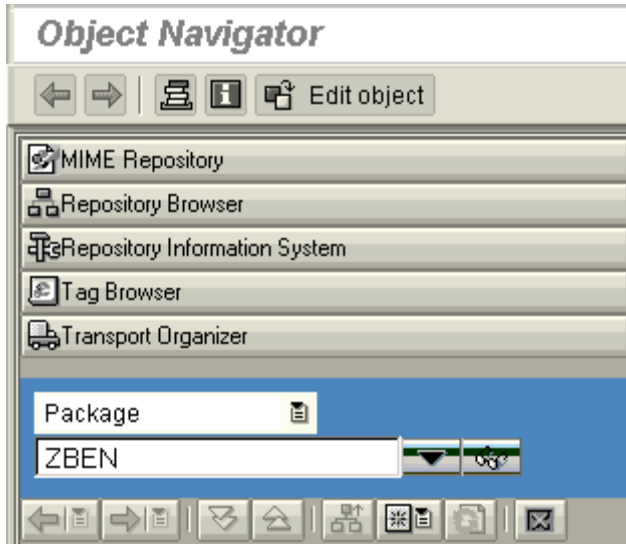
If the command line is hidden when you log on to the system, click the arrow highlighted below.




2. For releases prior to 4.7, select *Development Class* from the pull-down menu.



For release 4.7, select *Package* from the pull down menu.



3. Enter the four characters assigned to the Development Class or Package for the installation of the Adapter for SAP. For example, ZBEN.
4. Click *Display* (The button with the picture of glasses). 
The system responds: Development Class or Package ZBEN does not exist.
5. Click *Yes* to create the object.
6. Under *Create Development Class or Package* in the Short Text field, enter IBI-Created.
7. Accept all displayed defaults and click *Create*.
The system generates a unique CTS request #.
8. Save this CTS request # for future reference.
9. Click *Continue*.

Procedure How to Create a Function Group

1. Execute transaction SE80 to start the Object Navigator.
2. From the pull-down menu, select *Function Group*.
3. Enter the same four characters assigned to the Development Class or Package for installation of the Adapter for SAP. For example, ZBEN.
4. Click *Display*.
The system responds: Function Group ZBEN does not exist.
5. Click *Yes* to create the object.
6. Enter IBI-Created in the Short Text field under Create Function Group.
7. Accept all displayed defaults and click *Save* (unless otherwise directed by SAP Basis Administrator).
8. Under Create Object Directory Entry, enter the same four characters assigned to the Development Class, and click *Save*.
9. Under Prompt for transportable change request, enter the CTS request # assigned during the creation of the Development Class or Package.
10. Click *Continue*.

Procedure How to Deactivate the Unicode Checks

This procedure applies only to SAP R/3 release 4.7

1. Execute transaction SE80 to start the Object Navigator.
2. From the pull-down menu, select Function Group.
3. For Function Group value, enter the name of the Function Group that you created. For example, ZBEN.
4. Click *Display*.

The object window shows an open folder with the name of the function group and a closed folder named Includes.

5. Right-click the function group name, for example, ZBEN, and select *Change* from the pop-up menu.

The screenshot shows the 'Change Function Group' dialog box with the following fields and values:

Function group	ZBEN
Short text	For documentation
Person Responsible	BYAVUZ
Package	ZBEN
Application	S
Status	Activated
Program status	

Checkboxes:

- Editor lock
- Fixed point arithmetic
- Unicode checks active

Buttons at the bottom: Save, Change Requests (Organizer), Main program, Function group doc., and a close button (X).

6. Deselect the check box for Unicode checks active, and click *Save*.

Procedure How to Activate a Function Group

1. Execute transaction SE80 to start the Object Navigator.
2. From the pull-down menu, select *Function Group*.
3. For Function Group value, enter the name of the function group you created. For example, ZBEN.
4. Click *Display*.

The object window shows an open folder with the name of the function group and a closed folder named Includes.

5. Right-click the function group name, in this example ZBEN, and select *Activate* from the pop-up menu.

A new window opens which lists all inactive objects. Note that the object name now begins with SAPL.

Make sure the object for the function group. In the example, ZBEN is selected.

6. Click *Continue*.

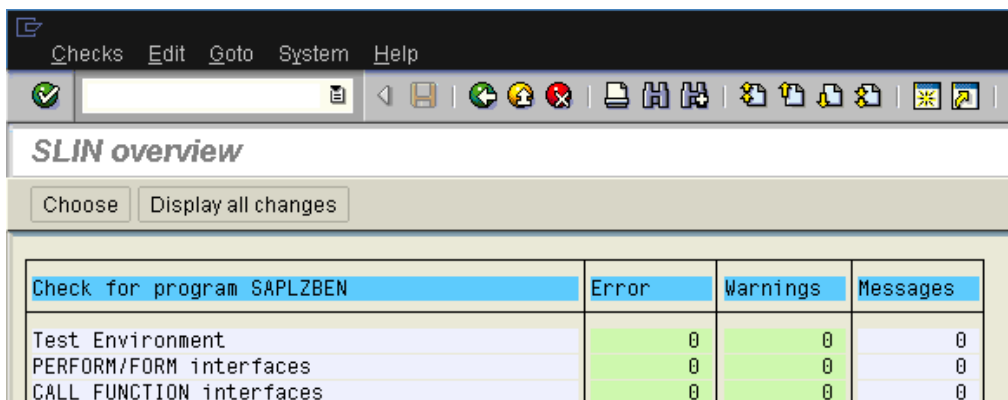
D..Object	Obj. name	User
REPS	SAPLZBEN	BYAVUZ

Procedure How to Verify a Function Group

1. Execute transaction SE80 to start the Object Navigator.
2. From the pull-down menu, select *Function Group*.
3. For Function Group value, enter the name of the function group created. For example, ZBEN.
4. Click *Display*.
5. Right-click the ZBEN folder and select *Check - Extended Check*.

6. Select *Perform Check*.

You should get a report with 0 errors, 0 warnings, 0 messages.

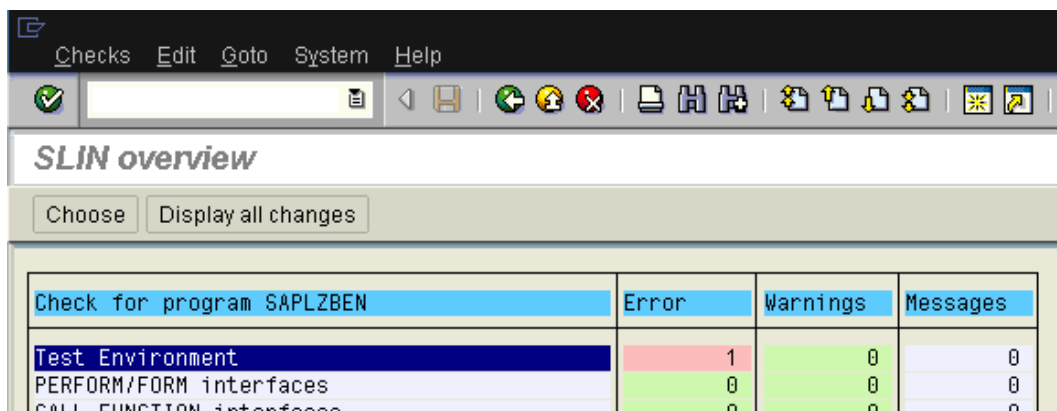


The screenshot shows the 'SLIN overview' window with a menu bar (Checks, Edit, Goto, System, Help) and a toolbar. Below the title bar are buttons for 'Choose' and 'Display all changes'. The main area contains a table with the following data:

Check for program SAPLZBEN	Error	Warnings	Messages
Test Environment	0	0	0
PERFORM/FORM interfaces	0	0	0
CALL FUNCTION interfaces	0	0	0

Note: If you encounter errors or warnings during this step, resolve them at the SAP R/3 layer before continuing with the installation of the Adapter for SAP.

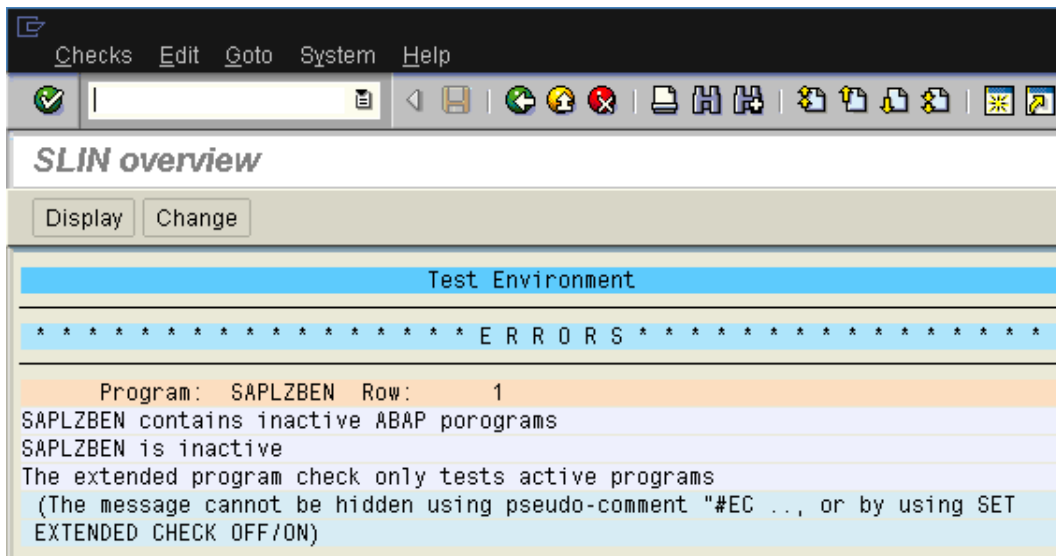
In the following example, the function group is not intentionally activated. The extended check returns an error.



The screenshot shows the 'SLIN overview' window with a menu bar (Checks, Edit, Goto, System, Help) and a toolbar. Below the title bar are buttons for 'Choose' and 'Display all changes'. The main area contains a table with the following data:

Check for program SAPLZBEN	Error	Warnings	Messages
Test Environment	1	0	0
PERFORM/FORM interfaces	0	0	0
CALL FUNCTION interfaces	0	0	0

Double-click the error line for detailed information about the error.



The SAP environment is now prepared for the next step of Function Module upload from the Web Console.

Accessing Multiple SAP R/3 Systems

The data adapter can operate across multiple R/3 systems. In R/3, data from multiple companies can share the same metadata, and is identified by a client value. Normally, a given system has multiple clients, and you may access one or more of them depending on your authorization. For each system, the data adapter requires one R/3 logon consisting of a client, a user, and a password. This logon must be:

- RFC enabled.
- Authorized for all clients that you may access.

You may need additional authorizations depending on the SAP data you wish to access. Consult your SAP administrator for details.

Configuring the Data Adapter for SAP R/3

Configuring the Data Adapter for SAP R/3 consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to SAP R/3, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Manually add the command in the global server profile (edasprof.prf) or in a user profile (user.prf).
- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the global server profile.

You can declare connections to more than one SAP R/3 data source by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to SAP R/3 takes place when the first query that references the connection is issued. If more than one SET CONNECTION_ATTRIBUTES command contains the same system, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Syntax

How to Declare Connection Attributes Manually

Explicit. The user IDs and password are specified for each connection and passed to SAP R/3 for authentication and request execution.

```
ENGINE [SQLSAP] SET CONNECTION_ATTRIBUTES  
system/user_ID,password:'client'
```

Password Passthru. The user ID and password received from the client application are passed to SAP R/3 for authentication and request execution.

```
ENGINE [SQLSAP] SET CONNECTION_ATTRIBUTES system/:'client'
```

where:

SQLSAP

Indicates the Data Adapter for SAP R/3. You can omit this value if you previously issued the SET SQLENGINE command.

system

Is the SAP System ID. It must point to a valid system entry in etc/sapserv.cfg.

user_ID

Is the SAP user ID for the user logon.

password

Is the password for the user logon.

Note: To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

client

Is the SAP client for the user logon. This value must be enclosed in single quotation marks (').

Procedure How to Declare Connection Attributes from the Web Console

The configuration page prompts you for the following connection attributes. The settings are stored in etc/sapserv.cfg. The information is stored by blocks. A block is identified by the system ID (usually the three character name of the SAP instance).

1. Launch the Web Console.
2. In the navigation pane, click Adapters.
The Configuring Data Adapters page opens.
3. In the left pane of the Configuration page, expand the Add Folder, then the SAP R/3 folder and click SAP R/3.

The Add SAP System Connect Parameters page opens.

User Authentication Parameters

The major work of the adapter is to translate user requests into code that can be understood by SAP. For this purpose, an SAP user ID, with a given set of privileges, is required. In the following table, this user ID is referred to as IBI_USER. After the request has been generated and is ready for execution, it can be executed either under the IBI_USER ID, or under a less privileged user (referred to as USER); results are then based on the user's credentials.

Security	<p>There are two methods by which a user can be authenticated when connecting to an SAP R/3 instance:</p> <p>Explicit. The user IDs and password are specified for each connection and passed to SAP R/3 for authentication and request execution.</p> <p>Password Passthru. The user ID and password received from the client application are passed to SAP R/3 for authentication and to execute the user's request.</p>
System	Is the SAP system ID.
IBI_CLIENT	Is the SAP Client for the IBI logon, maximum 3 characters.

CLIENT	Is the SAP Client for the user logon, maximum 3 characters.
IBI_USER	Is the SAP user ID for the IBI logon.
USER	Is the SAP user ID for the user logon.
IBI_PASSWORD	Is the SAP password for the IBI logon, maximum 8 characters.
PASSWORD	Is the SAP password for the user logon, maximum 8 characters.

Server Internal Parameters

FPOOL	Is the function group where the Adapter for SAP R/3 static function modules can be cataloged. This is the function group created in <i>Preparing the SAP R/3 Environment</i> on page 34-2.
DATACLASS	Is the SAP data class for user defined SAP tables generated by the Adapter for SAP R/3 (default appl0). Valid classes are appl0, appl1, appl2, and usr1. The maximum is 5 characters.
TMPDIR	Is the temporary directory to be used for SAP extracts (default /tmp/), maximum 64 characters. These extracts are flat files that can only be written to a read/write directory, and are readable for the end user.
RANGEBEG	Is the beginning range for the function group.
RANGEEND	Is the end range for the function group.

Connection Parameters

APPGRP	Is the name of the application group. An application group defines a list of application servers, on which an RFC application can be running. R/3 transaction SMLG can be used to view or modify application groups.
MSGHOST	Is the name of the SAP message server.
HOST	Is the host name of the SAP application server.
GWHOST	Is the host name of the machine where the SAP gateway process is running. In the case where there is only one SAP application server, gwghost and host is the same.
SYSNR	Is the SAP system number. Obtain this value from the SAP Administrator.
LANGUAGE	Is the language installed on SAP to be used with the Adapter for SAP. Typically this is the default SAP language.

Parameters for Mixed Character Code Set

The following parameters apply only when the server platform and the SAP instance platform do not use the same character code set (ASCII or EBCDIC). Please refer to *Supporting Mixed Code Page Environments* on page 34-35 for detailed information.

eda2sap	Is the server to SAP host conversion table, as generated in <i>Preparing the SAP R/3 Environment</i> on page 34-2. This is the conversion file where the server code page is the source and SAP code page is the target. In most cases, this parameter is blank.
sap2eda	Is the SAP host to server conversion table file name, as generated in <i>Preparing the SAP R/3 Environment</i> on page 34-2. This is the conversion file in which the SAP code page is the source and the server code page is the target. In most cases, this parameter is blank.

Note: If the SAP R/3 system is running on a MVS / USS environment, see *How to Configure the Adapter on a MVS or OS/390 and z/OS Environment* on page 34-15.

Once all necessary parameter values are entered, click *Configure* to save the entry in the `edasprof.prf` file.

After configuring the adapter, it is necessary to install the static function modules in the SAP data dictionary to complete the configuration.

Procedure How to Configure the Adapter on a MVS or OS/390 and z/OS Environment

To configure the query adapter if running on a MVS or OS/390 and z/OS environment, the SAP Code Page 0126 must be installed on the R/3 server. This code page will be provided with one of the upcoming SAP Basis Support Packages, possibly with 45. Follow the instructions from SAP to install the Code Page.

1. Create two conversion tables by following the instructions in *Supporting Mixed Code Page Environments* on page 34-35, and transfer the tables to the `$EDACONF/etc` directory.

For example, if the iWay server environment uses the code page 1100, then two conversion tables, `11000126.CDP` and `01261100.CDP`, should be created and transferred to the `$EDACONF/etc` directory.

2. Edit the edastart.bat file and add the following two lines:

```
export SAP_CODEPAGE=0126
```

```
export PATH_TO_CODEPAGE=$EDACONF/etc/
```

Note: Make sure to include the trailing "/" for the value of PATH_TO_CODEPAGE parameter.

3. Assign the correct codepage files to the parameters of eda2sap and sap2eda by using the Web Console or manually editing. Make sure to enter CDP in capital letters. For example:

```
eda2sap=01261100.CDP
```

```
sap2eda=11000126.CDP
```

Procedure How to Install SAP Components

To install SAP components:

1. Launch the Web Console.
2. In the navigation pane, click *Adapters*.
3. In the left pane, expand the Configured folder, then the SAP R/3 folder and click the system created in *How to Declare Connection Attributes from the Web Console* on page 34-13.

A pop-up menu opens.

4. Select *Properties*.

The Change SAP System Connect Parameters window opens in the right pane.

5. If you are re-installing due to an error in a previous installation, scroll to the bottom of the left pane, which shows the Install SAP Components button.
6. Select the check box "Overwrite existing Function Group" only if an error occurred in a previous installation. Click *Install SAP Components*.

A message appears confirming this request.

7. Click *OK* to continue the installation.
8. A new browser window opens with a message "Processing the request".

This process may take a few minutes. Do not close the browser window. The message Done appears in the browser window along with any errors.

The following is a sample error message which you may get this if you try to re-install the SAP components without checking the Overwrite Existing Function Group:

```
(FOC44427) YOU ARE ABOUT TO OVERWRITE THE FOLLOWING FUNCTION MODULES
(FOC44428)   ZY52_DYNAMIC_RUN
(FOC44428)   ZY52_GET_APPLICATION_TREE
(FOC44428)   ZY52_REPORT_ABORT_BATCH
(FOC44428)   ZY52_REPORT_CREATE
(FOC44428)   ZY52_REPORT_DELETE
(FOC44428)   ZY52_REPORT_GET_BATCH_DATA
(FOC44428)   ZY52_REPORT_GET_BATCH_STATUS
(FOC44428)   ZY52_REPORT_RUN
(FOC44428)   ZY52_REPORT_RUN_IN_BATCH
(FOC44428)   ZY52_TEST_REVERSE
(FOC44429) RERUN THE COMMAND WITH THE OVERWRITE OPTION
```

Procedure How to Verify Installation of SAP Components

To confirm that all necessary function modules are successfully installed:

1. Run transaction SE80.
2. Select Function Group from the first drop-down list.
3. Enter the name of the Function Group you created in *How to Declare Connection Attributes from the Web Console* on page 34-13.
4. Click *Display* next to the text box.

There folders named Function Modules and Includes should be under the Function Group name.

5. Expand the Function Modules and Includes folders to verify that the following appears:

```
Z***_DYNAMIC_RUN
Z***_GET_APPLICATION_TREE
Z***_REPORT_ABORT_BATCH
Z***_REPORT_CREATE
Z***_REPORT_DELETE
Z***_REPORT_GET_BATCH_DATA
Z***_REPORT_GET_BATCH_STATUS
Z***_REPORT_RUN
Z***_REPORT_RUN_IN_BATCH
Z***_TEST_REVERSE
```

Note: Z*** represents the name of the Function Group. If the Development Class and Function Group name is ZIBI, then the Function Module name begins with ZIBI.

Managing SAP R/3 Metadata

When the server accesses a data source, it needs to know how to interpret the data found. For each data source the server accesses, you create a synonym that describes the structure of the data source and maps the server data types to the SAP R/3 data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each SAP data object that is accessible from the server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File, which represent the server's metadata.

Reference Types of SAP Synonyms

The Data Adapter for SAP R/3 enables you to create:

- **SAP Synonyms**

To create synonyms for SAP tables, LDBs (Logical Data Base) or BAPIs (Business Application Programming Interface), the adapter requires that base metadata be available. You must generate this base metadata before you begin the synonym creation. This task is carried out using the same window for creating the synonyms. The task is submitted as a deferred request and it takes about 15-20 mins. It is suggested that you should not do any other activity using the web console while the base is being generated. If an underlying data object changes, you must regenerate base metadata for subsequent synonym creation.

- **FM Synonyms**

To create synonyms for SAP function modules, the only thing needed is to know the exact name of the function module. Please note that not all function modules are suitable for reporting.

- **IDOC Synonyms**

To create synonyms for IDOCs (Intermediate Document Interface), the adapter requires that an IDOCs list be available. You must generate this list before you begin the synonym creation for the first time. If an underlying IDOC changes, you must recreate the IDOC list for subsequent synonym creation. Please keep in mind that IDOC synonyms are used for ETL purposes and not for queries.

- **Query Synonyms**

An SAP Query is a predefined report designed for users with little or no knowledge of the SAP programming language ABAP. From InfoSets (data sources), which are organized in user groups, an SAP query creates a list of information not already contained in the SAP system. Being a report, it may have variants. A variant is a saved set of selection criteria for a query or the report generated by the query. If you specify a variant when starting a query, the system uses the values in the variant for the query selection criteria.

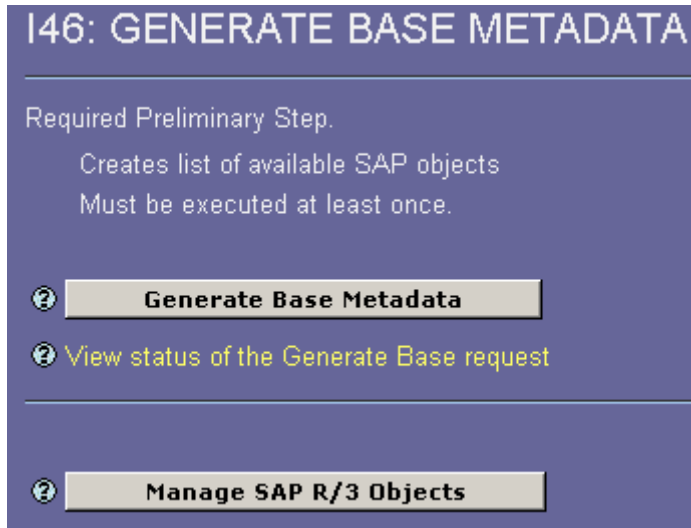
To create a synonym for an SAP Query, base metadata must be created. Note that the SAP Query base metadata must be refreshed to see newly created queries.

Procedure **How to Create an SAP Synonym Using the Web Console**

1. Start the Web Console. In the navigation pane, click *Metadata*. The Editing Metadata page opens.

To create a synonym, you must have configured the adapter. For information on how to configure an adapter, see *Configuring the Data Adapter for SAP R/3* on page 34-12.

2. Expand the Add folder and then the adapter folder, and click a connection.
3. From the resulting menu, select *Create SAP Synonym*. The Generate Base page opens:



4. If necessary, generate base metadata. For more information, see *Types of SAP Synonyms* on page 34-18.

- Click *Manage SAP R/3 Objects*.

The Synonym Management pane appears:

I46: Synonym Management

Synonym type:
 TABLE
 LDB
 BAPI

Filter by TABLE / LDB / BAPI Name

Name - Sample: abc* **Select**

Filter by SAP R/3 Application

- Select a synonym type, either Table, LDB, or BAPI.
- To filter by name, in the Synonym Name field, enter a string, using the wildcard character (*) as needed
- Click *Select by Name*.

To filter by application, click *Filter by SAP R/3 Application*. Select an application, and then click *Select by Application*.

Synonym type:
 TABLE
 LDB
 BAPI

Filter by TABLE / LDB / BAPI Name

Filter by SAP R/3 Application

Select Application

Select	Application Key	Application Label
<input type="radio"/>	ALR0000021	Accounting - General
<input type="radio"/>	ALR0000081	Customer Service
<input type="radio"/>	ALR0000091	Logistics Execution
<input type="radio"/>	B200000092	Service

The table names that meet your criteria are displayed.

Note: If you receive the message "No Tables Found," it is likely that base metadata was not generated. See Step 4.

9. From the Switch Current Application Directory drop-down list, select a directory. The default value is baseapp.
10. Select the tables for which you want to create synonyms and click *Create Synonym*. Synonyms are created and added under the specified application directory.

Procedure How to Create an IDOC Synonym Using the Web Console

1. Start the Web Console. In the navigation pane, click *Metadata*. The Editing Metadata page opens.

To create a synonym, you must have configured the adapter. For information on how to configure an adapter, see *Configuring the Data Adapter for SAP R/3* on page 34-12.

2. Expand the Add folder and the adapter folder, and click a connection.
3. From the resulting menu, select *Create IDOC Synonym*. The IDOC Metadata Management page opens:

I46: FUNCTION MODULE METADATA MANAGEMENT

Select Application: baseapp

FM Name Entry_Exact_FM_Name - Sample: g_set_tree_import

Create FM Synonym

4. If necessary, create an SAP IDOCs list. For more information, see *Types of SAP Synonyms* on page 34-18.
5. To list IDOCs for a specific message type, enter a message type in the Message Type field; for example, MATMAS.

To list all IDOCs, leave the Message Type field blank.

6. Click *Select IDOC Synonyms*. The IDOCs that meet your criteria are listed.

Select	Message Type	IDOC Type	Extention	Description	Release
<input type="checkbox"/>	MATMAS01	MATMAS		Material master	30A
<input type="checkbox"/>	MATMAS02	MATMAS		Material master	30D
<input checked="" type="checkbox"/>	MATMAS03	MATMAS		Material master	40A

Note: If you receive the message "No IDOCs Found," it is likely that the IDOC list was not generated. See Step 4.

7. Select a directory from the Switch Current Application Directory drop-down list. The default value is baseapp.
8. Ensure that Replication is checked.
9. Select the IDOCs for which you want to create synonyms, and click *Create IDOC Synonym*. Synonyms are created and added under the specified application directory.

Procedure How to Create an FM Synonym Using the Web Console

1. Start the Web Console. In the navigation pane, click *Metadata*. The Editing Metadata page opens.

To create a synonym, you must have configured the adapter. For information on how to configure an adapter, see *Configuring the Data Adapter for SAP R/3* on page 34-12.

2. Expand the Add folder and then the adapter folder, and click a connection.
3. From the resulting menu, select *Create FM Synonym*.

The Function Module Management page opens:

I46: FUNCTION MODULE METADATA MANAGEMENT

Select Application: baseapp

FM Name Entry_Exact_FM_Name - Sample: g_set_tree_import

Create FM Synonym

4. From the Switch Current Application Directory drop-down list, select a directory. The default value is baseapp.
5. In the Function Module name field, enter a module name and click *Create FM Synonym*. The synonym is created and added under the specified application directory.

Procedure **How to Create a SAP Query Synonym Using the Web Console**

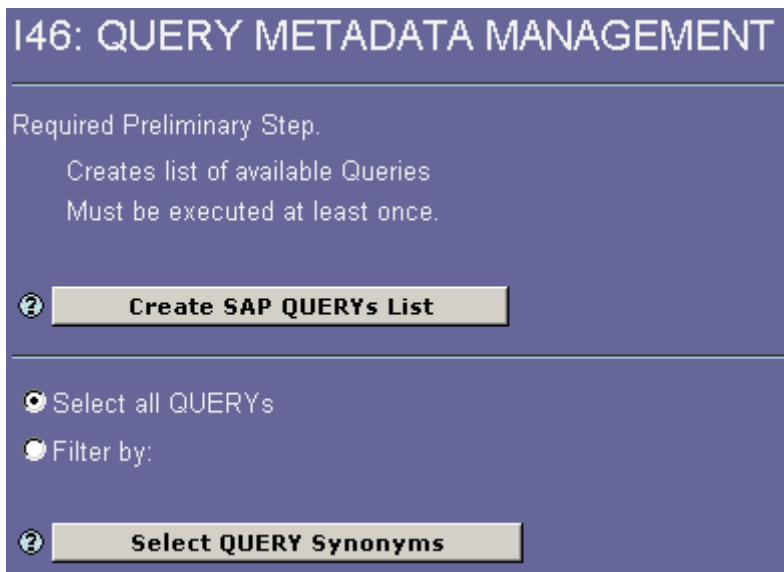
1. Start the Web Console and, in the navigation pane, click *Metadata*.

The Editing Metadata page opens.

To create a synonym, you must have configured the adapter. For information on how to configure an adapter, see *Configuring the Data Adapter for SAP R/3* on page 34-12.

2. Expand the Add folder, expand the adapter folder, and then click a connection.
3. From the resulting menu, select *Create SAP Query Synonym*.

The Query Metadata Management page opens:



4. If necessary, select *Create SAP QUERYs List*. For more information, see *Types of SAP Synonyms*.
5. To list QUERYs for a specific Group, select *Filter by* and enter a Group Name. You can also enter the Query Name to filter the result set.

To list all QUERYs, choose *Select all QUERYs*.


```
FILE=BASEAPP/ALM01_02 ,SUFFIX=SQLSAP,
REMARKS='UG: 02 Query: ALM01', $
SEGNAME=VARIANT, SEGTYPE=S0, $
FIELD=VARIANT, ALIAS=VARIANT, ACTUAL=A14, USAGE=A14, ACCEPT=
'CAUS' OR
'MY VARIANT' OR
', $
SEGNAME=REPORT, PARENT=VARIANT, SEGTYPE=S0, $
FIELD=SKA1_SAKNR, ALIAS=SKA1_SAKNR, USAGE=A010, ACTUAL=A010,
TITLE='G/L acct', DESC='G/L account number', $
FIELD=SKA1_GVTYP, ALIAS=SKA1_GVTYP, USAGE=A002, ACTUAL=A002,
TITLE='AT', DESC='P&L statement account type', $
FIELD=SKC1C_BUKRS, ALIAS=SKC1C_BUKRS, USAGE=A004, ACTUAL=A004,
TITLE='CoCd', DESC='Company Code', $
FIELD=SKC1C_GJAHR, ALIAS=SKC1C_GJAHR, USAGE=A004, ACTUAL=A004,
TITLE='Year', DESC='Fiscal year', $
FIELD=SKC1C_GSBER, ALIAS=SKC1C_GSBER, USAGE=A004, ACTUAL=A004,
TITLE='BA', DESC='Business Area', $
FIELD=SKC1C_HWAER, ALIAS=SKC1C_HWAER, USAGE=A005, ACTUAL=A005,
TITLE='Crcy', DESC='Currency Key', $
FIELD=SKC1C_UM010, ALIAS=SKC1C_UM010, USAGE=P16.2, ACTUAL=A16,
TITLE='SKC1C-UM01', DESC='Monthly balance', $
FIELD=SKC1C_HWAER1, ALIAS=SKC1C_HWAER1, USAGE=A005, ACTUAL=A005,
TITLE='Crcy', DESC='Currency Key', $
```

A valid request using variant 'CAUS' is:

```
TABLE FILE ALM01_02
PRINT
  SKA1_SAKNR
  SKA1_GVTYP
  SKC1C_BUKRS
  SKC1C_GJAHR
  SKC1C_GSBER
  SKC1C_HWAER
  SKC1C_UM010
  SKC1C_HWAER1
IF VARIANT EQ 'CAUS'
IF READLIMIT EQ 50
END
```


Reference **Managing Synonyms**

In the navigation pane, click the name of the synonym to access the following options:

Sample Data	Retrieves up to 20 rows from the associated data source. Note: This option is not available for IDOC synonyms.
Properties	Displays a graphic representation of the synonym and enables you to edit its metadata.
Edit Master File	Enables you to manually edit the synonym's Master File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Master File, to update the synonym.
Edit Access File	Enables you to manually edit the synonym's Access File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Access File, to update the synonym.
Refresh	Regenerates an SAP or FM synonym. Use this option if structural changes were made to the data source.
Archive	Regenerates an IDOC synonym. Use this option if structural changes were made to the source IDOC.
Drop	Deletes the synonym.
Copy to	Copies the synonym to another application directory. Click the target directory from the resulting list.
Move to	Moves the synonym to another application directory. Click the target directory from the resulting list.

Syntax **How to Create a Synonym Manually**

```
CREATE SYNONYM appname/synonym FOR source DBMS SQLSAP AT system  
END
```

where:

appname

Is the 1- to 64-character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used. If your server is not APP enabled, this application name must not be used.

synonym

Is an alias for the data source (maximum 64 characters).

source

Possible values are: TABLE, LDB, BAPI, FM, IDOC, QUERY.

SQLSAP

Indicates the Query Adapter for SAP.

system

Is a system name previously defined using the SET CONNECTION_ATTRIBUTES command. When the synonym is created, this system name becomes the value for the CONNECTION attribute in the Access File.

END

Indicates the end of the command, and must be on a separate line in the procedure.

SAP R/3 Table Support

The following table describes support for R/3 tables:

R/3 Table Type	Server Support
TRANSPARENT	Supported.
CLUSTER TYPE 1	Type 1 (for example, BSEG). This type of cluster has only one metadata definition (or set of fields), and can be read using Open/SQL statements. The data adapter supports this type of cluster.
CLUSTER TYPE 2	Type 2 (a repository, for example, Payroll in HR). In the same physical file, R/3 stores metadata and data. Open/SQL cannot be used to read these clusters. They must be accessed by a specific ABAP program, using R/3 internal macros. The data adapter does not support this type of cluster.
CLUSTER TYPE 3	Type 3 (an in-core cluster, for example, Cost Accounting Hierarchy). An R/3-specific program must be written to access these clusters. The data adapter does not support this type of cluster.
POOL	Supported.
VIEW	Supported if composed of TRANSPARENT, POOL or accessible CLUSTER tables.

SAP R/3 Data Type Support

The following table lists the R/3 data types and notes how they are mapped to data types in the Master File.

R/3 Data type	R/3 Length	R/3 Decimals	Usage	Actual
ACCP	N	0	An	An
CHAR	N	0	An	An
CLNT	3	0	A3	A3
CUKY	N	0	An	An
CURR	n	p	P(n+2).p	P(n+1)/2
DATS*	8	0	YYMD or DMY	A8
DEC	n	p	P(n+2).p	P(n+1)/2
FLTP	n	p	P(n+2).p	P(n+1)/2
INT1	3	0	I3 (limited to 127)	I1
INT2	5	0	I4	I2
INT3	5	0	I4	I3
INT4	10	0	P12.0	P6
LANG	1	0	A1	A1
LCHR	N	n	not supported	
LRAW	N	N	not supported	
NUMC	N	0	An	An
PREC	N	N	not supported	
QUAN	N	P	P(n+2).p	P(n+1)/2
RAW	N	n	not supported	
TIMS	6	0	A6	A6 (00:00:00)
VARC	N	0	An (<=255)	An (<=255)
UNIT	3	0	A3	A3

* The date usage (DATS) is dynamically determined based on the value of the country supplied in the general server configuration. Therefore, date presentations will vary by country.

SAP R/3 Open/SQL Support

The following table describes the conversion that takes place from the FOCUS statements to SAP's Open/SQL statements.

FOCUS Command	Corresponding Open/SQL statement
READLIMIT	UP TO <i>n</i> ROWS When a multiple SELECT statement is generated. READLIMIT is translated in UP TO <i>n</i> ROWS for each individual SELECT.
WHERE	It is generally translated into a WHERE statement. However, in some cases, due to SAP's SQL buffer limit it might be translated into a CHECK command of ABAP/4.
JOIN TO	For each table included in the JOIN command, embedded SELECT statements are created.
JOIN	It is translated into SELECT ... SELECT SINGLE ...
NULL	The Data Adapter for SAP R/3 does not support the use of the reserved word 'NULL' as part of a WHERE clause. Including 'NULL' as part of a WHERE clause results in a syntax error.
SUM ... BY	By default the aggregation feature is turned off. If the aggregation is turned off individual rows are selected and aggregation and sorting is done by FOCUS. It can be turned on by issuing SQL SQLSAP SET OPTIMIZATION SQL Once the aggregation is turned on the SUM is translated into SUM and the BY is translated into GROUP BY and ORDER BY. Aggregation feature only applies to TRANSPARENT tables.

Advanced SAP R/3 Features

This topic describes how to use security, BAPIs, joins, and tracing.

Support for User Security

The Native interface for SAP R/3 has been enhanced to handle the user ID and password provided in

```
SQL SQLSAP SET CONNECTION_ATTRIBUTES system/user,password:'client'
```

This feature allows you to run a secured request that involves the following:

- One or more BAPIs, as BAPIs are secured by SAP.
- SAP delivered logical databases, as the server code is secured by SAP.
- Function Modules (either SAP or customer) that are fully prototyped, including the proper AUTHORITY-CHECK statements.

BAPI Support

This release supports most read-only BAPIs, including joins, as described in the following example:

```
CREATE SYNONYM baseapp/BUS0002_GETLIST
FOR BUS0002/GETLIST
BAPI DBMS SQLSAP AT I46
END
CREATE SYNONYM baseapp/BUS0002_GETDETAIL
FOR BUS0002/GETDETAIL BAPI
DBMS SQLSAP AT I46
END
JOIN BAPI0002_COMP_CODE IN COMPANYCODE_GETLIST TO
CCGD2_COMP_CODE IN OMPANYCODE_GETDETAIL
END
TABLE FILE COMPANYCODE_GETLIST
PRINT
CCGL0_TYPE NOPRINT
BAPI0002_COMP_CODE
BAPI0002_COMP_NAME
CCGD2_CURRENCY
CCGD2_LANGU
IF BAPI0002_COMP_CODE NE '2300' OR '6000'
END
```

Joins Support

The Native Interface supports all joins from and to SAP. For performance reasons, we do not recommend joining to an SAP data source from a non-SAP data source. It is more efficient to hold the keys in a sequential file, and then use the following code:

```
TABLE FILE SAP PRINT FIELDS IF KEYS IS (HOLD) END
```

Setting up the Report Processing Mode

SAP R/3 supports many different types of processes (for example, batch, dialog, print, and so on). Each WebFOCUS for SAP query creates a process within the R/3 application server. This process can execute as either a dialog or batch process.

These processing mode options allow you to adapt to the R/3 application server configuration where there are a specific number of processes allocated for each process type per R/3 application server.

Dialog Process

Dialog processing is the default mode for the SAP R/3 Data Adapter. An SAP dialog process is suitable for a task that needs to run immediately at a higher priority. An SAP dialog process is subject to an idle timeout limit where the default value is 15 minutes. Check with your SAP Basis consultant to see if this default value has changed. Dialog process mode should be set for reports that have processing times below the idle timeout limit. If a report's processing time exceeds the dialog process idle timeout limit, the SAP R/3 application server terminates the process and the report fails to run.

If you have a WebFOCUS for SAP report that requires longer processing time, see *Batch Mode Processing* on page 34-35.

The following command sets your SAP R/3 Adapter to dialog processing mode:

```
SQL SQLSAP SET EXECUTIONMODE ONLINE
```

Note: This command can be inserted into the EDA Server profile (for example, edasprof.prf) or included within the report procedure.

You would use this command if you had explicitly set batch execution mode and wanted to switch back to dialog mode within a single session on the EDA Server.

Batch Mode Processing

Batch processing is the other mode for the SAP R/3 Data Adapter. An SAP batch process is suitable for a task that can afford to run in the background at a lower priority. This mode is for reports that require longer processing time (for example, reports that involve joining multiple tables or month end reports that need to process a lot of data).

The following command will set your SAP R/3 Adapter to batch processing mode:

```
SQL SQLSAP SET EXECUTIONMODE BATCH
```

Note: This command can be inserted into the EDA Server profile (for example, edasprof.prf) or included within the report procedure.

Supporting Mixed Code Page Environments

SAP R/3, as well as adapters, can be installed on various platforms. If SAP R/3 is installed on a platform that uses a code page or character set different from the adapter platform, two character conversion tables must be generated to translate the different character sets, one for each direction of data flow.

For example, SAP R/3 is installed on an OS/400 EBCDIC machine and the SAP R/3 data adapter is installed on an Intel Windows ASCII machine. If a request is going from the server to SAP R/3, an ASCII to EBCDIC translation is required. If SAP R/3 is sending data back to the SAP adapter, an EBCDIC to ASCII translation is required. This section describes how to create conversion tables and where they must reside.

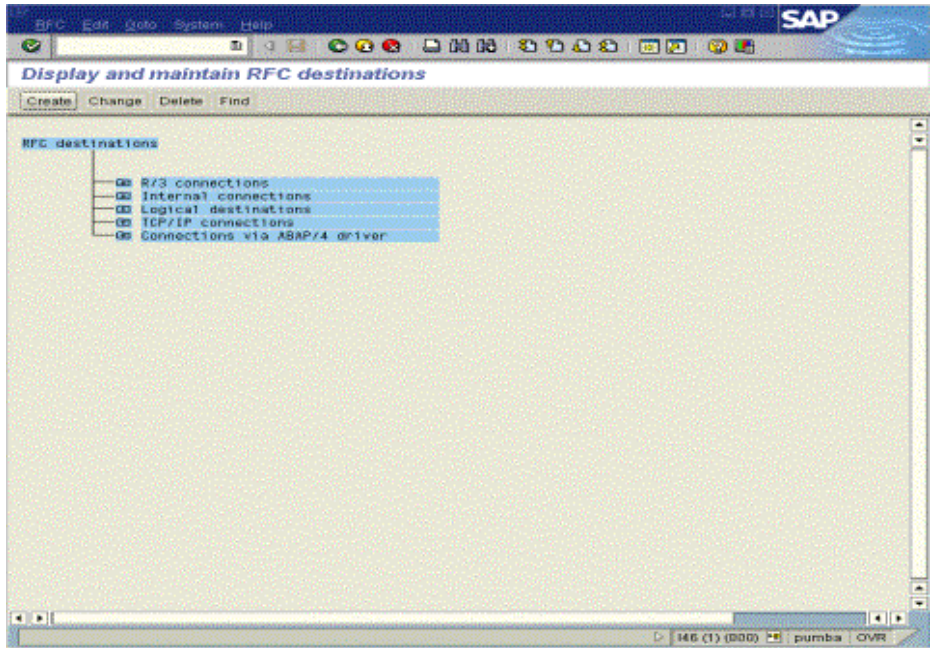
Character Conversion Tables

SAP R/3 provides a transaction, SM59, to generate conversion tables for RFC based applications such as the SAP R/3 adapter. Once these tables are created, they must be copied to the adapter.

Procedure **How to Generate Conversion Tables**

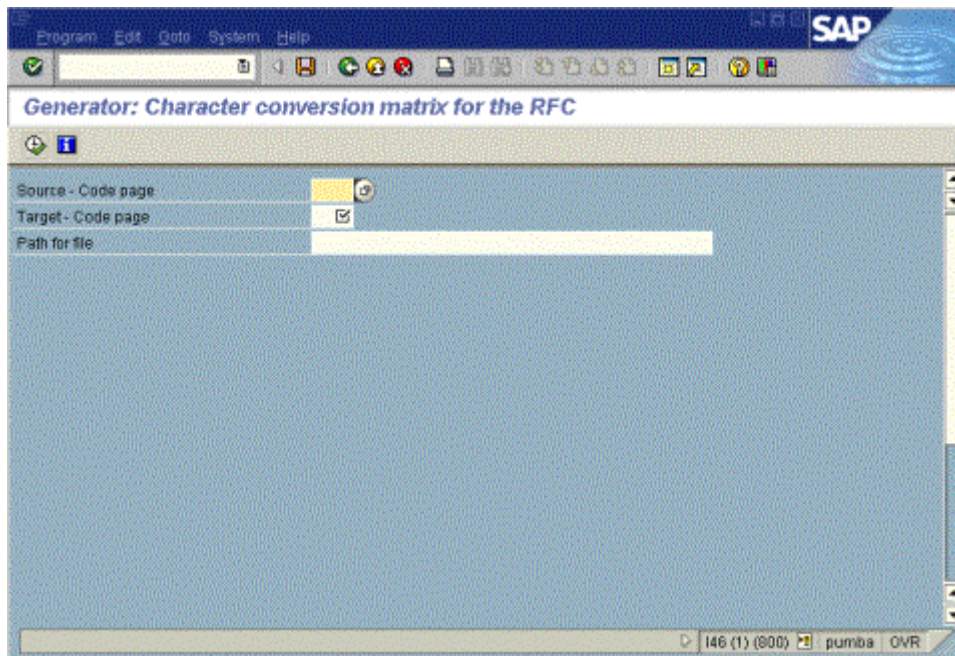
1. Enter transaction SM59 at the SAPGUI command line.


The following screen displays:



2. Select the Generate Conv. Tab option from the RFC pull-down menu.

The following screen displays:



3. Enter a value for the Source code page, Target code page and a valid path on the R/3 application server where the conversion table file is created. Optionally you can supply a valid path and a unique file name.
4. Click the Execute button .

If you did not specify a unique file name, the default file name of the newly created conversion table will contain the source and target code page values. For example, if you specified a source code page of 0102 (OS/400 for some CUAs) and a target code page of 1101 (7 bit USA ASCII pur), the default file name is 01021101.cdp. This is your OS/400 (EBCDIC) to ASCII conversion table file.

It is now necessary to create another conversion table file for translating code pages in the opposite direction.

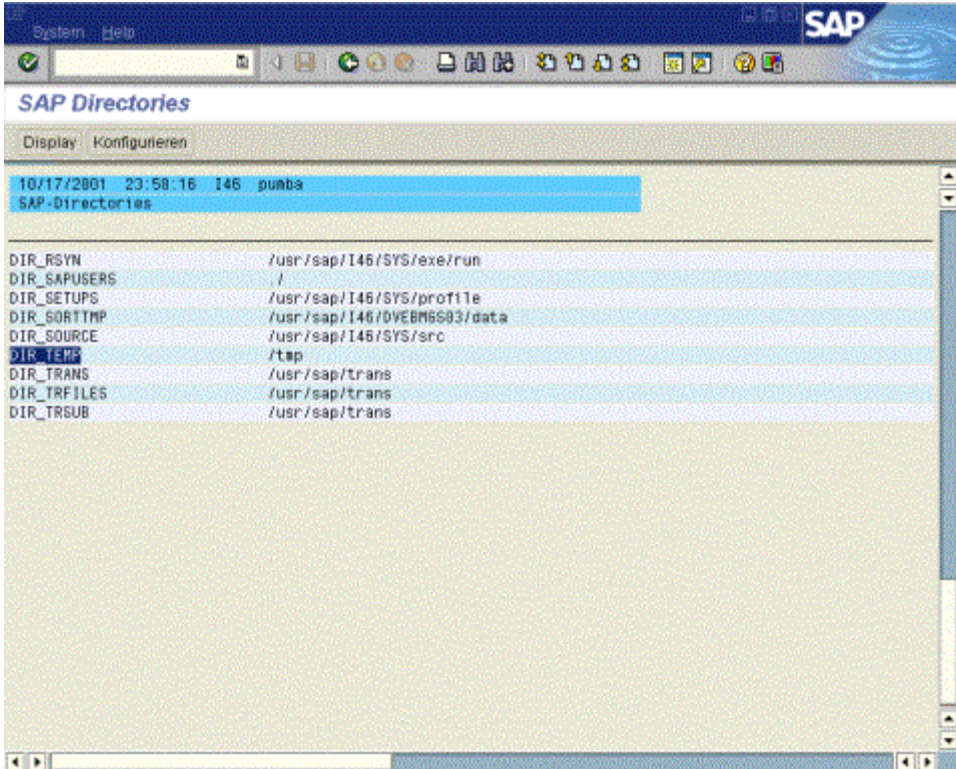
5. Switch the source and target code pages and generate the conversion table.

Continuing with previous example, you should have a source code page of 1101 and a target code page of 0102, resulting in a default file name of 11010102.cdp. This is your ASCII to OS/400 (EBCDIC) conversion table file.

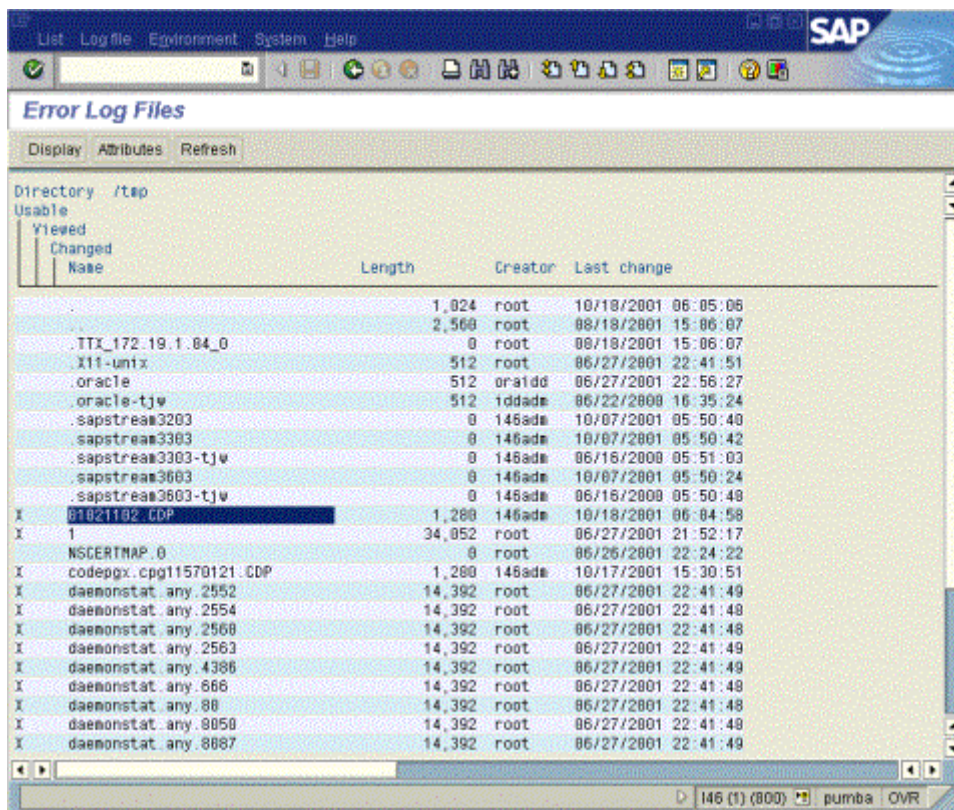
Note: If a unique file name was used for the first conversion table, be sure you specify a unique file name for the second conversion table.

Procedure **How to Download the Conversion Tables to the Server**

Once these conversion table files are created you can download them from the SAP application server directory to the server configuration ETC directory. Use SAP R/3 transaction AL11 to browse and download the two conversion table files to a local file on the desktop or Windows machine where the server is installed. Note that SAPGUI is required on the machine to which you are downloading the conversion table files. The following screen represents transaction AL11:

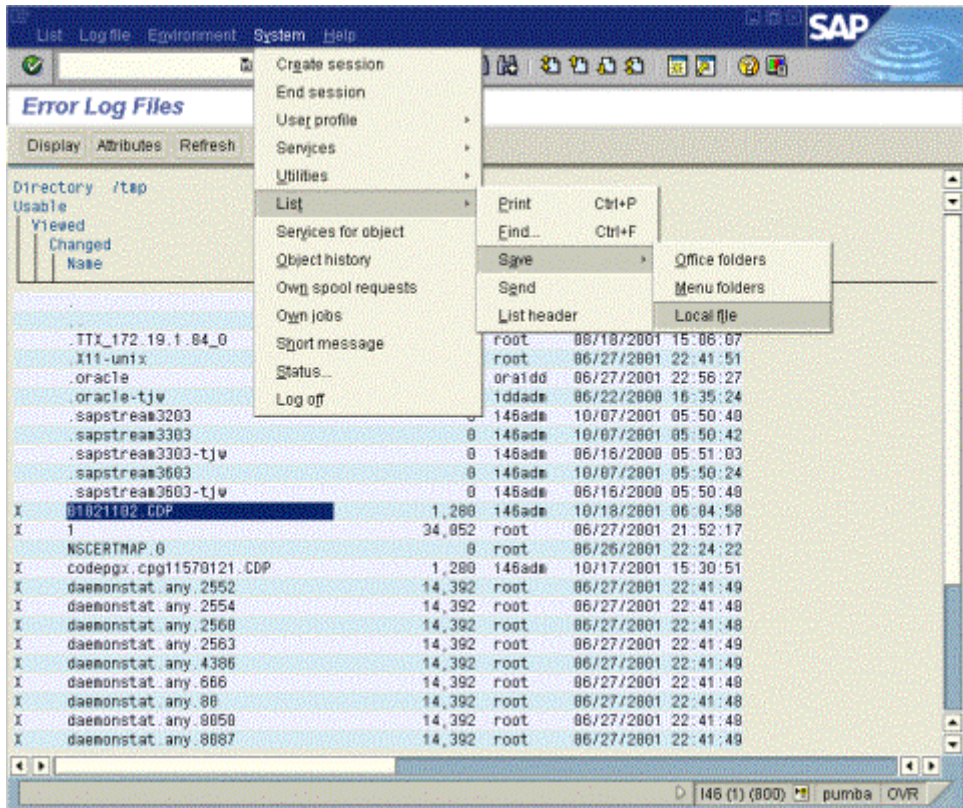


1. Double-click the directory where the conversion table files were created.
The following window displays:



2. Double-click the conversion table file to view its contents.

3. Select the List option from the System pull-down menu and select *Save to Local File* as illustrated in the following screen:



A dialog box displays prompting on a format for the transfer file.

4. Select unconverted format and click *Continue*.
5. Enter a valid file name and click *Transfer*. This transfers the file to your local desktop.
6. Use any available editor on your desktop and remove the first three lines from the conversion table file. The following is an example of the text that should be removed:

```
Directory /tmp
Name: 01021102.CDP
-----
```

After the edits have been made, configure the server with the SAP R/3 adapter as described in *Configuring the Server*. Once the SAP R/3 adapter is configured, the conversion table files must be moved to the server's EDHOME ETC directory. Make sure both conversion table files are moved.

Tracing Options

Traces are enabled from the Web Console. Use Custom option and select one of the following:

- SQLCALL
- SQLSAP/1
- SQLSAP/2
- SQLSAP/3

where:

level

Indicates the trace level. Currently supported trace levels are:

- Generated ABAP/4 program.
- Retrieved segments information.

You may also run a diagnostic trace from the SAP system by running transaction ST05 (Performance Trace). See your SAP system Administrator for details.

Producing SAP R/3 Requests

The following requests and output are examples of the capabilities of the SAP R/3 query adapter using standard ANSI SQL code. These examples require that CREATE SYNONYM has been performed on the relevant tables from the Web Console. You can use the Web Console to navigate the SAP application hierarchy by searching the relevant tables or by doing a simple search by name.

Example Retrieving All Records

The following syntax retrieves all records from T014:

```
SELECT * FROM T014;
```

WebFOCUS equivalent of this SQL statement is:

```
TABLE FILE T014
PRINT *
END
```

The output is:

Client	Credit Area	Currency	Update	FYI variant	Risk Categ.	Cred. Limit	Rep. group	All CoCdes
800	0001	EUR	000012			.00		
800	1000	EUR	000012	K4		.00		
800	3000	USD	000012	K4		.00		
800	5000	JPY	000012	K4		.00		
800	6000	MXN	000012	K4		.00		
800	R100	EUR	000012	K4		.00		
800	R300	USD	000012	K4		.00		

Example Retrieving Selected Fields

The following syntax retrieves selected fields from MARA:

```
SELECT MARA_MATNR, MARA_MTART, MARA_MATKL, MARA_WRKST, MARA_BISMT FROM
MARA WHERE MARA_MATKL='999' ;
```

WebFOCUS equivalent of this SQL statement is:

```
TABLE FILE MARA
PRINT MARA_MATNR MARA_MTART MARA_MATKL MARA_WRKST MARA_BISMT
WHERE MARA_MATKL EQ '999' ;
END
```

The output is:

Material	Matl_type	Matl_group	Basic_matl	Matl_no_
00000000000000000038	HALB	999		
AM3-GT	KMAT	999		
BG100001	HALB	999		
KR090654	HALB	999		

Example **Joining Two Tables**

The following syntax joins the MARA database with the MARC database:

```
SELECT MARA_MATNR, MARA_WRKST, MARC_WERKS FROM MARA, MARC WHERE
MARA_MATNR=MARC_MATNR AND MARC_WERKS='5100';
```

WebFOCUS equivalent of this SQL statement is:

```
JOIN CLEAR *
JOIN MARA_MATNR IN MARA TO ALL MARC_MATNR IN MARC AS JO
TABLE FILE MARA
PRINT MARA_MATNR MARA_WRKST MARC_WERKS
WHERE MARC_WERKS EQ '5100';
END
```

The output is:

Material	Basic_matl	Plant
NC-100-212		5100
NC-100-213		5100
NC-100-311		5100
NC-100-312		5100
NC-103-101		5100
NC-103-211		5100

CHAPTER 35

Getting Started in Siebel

Topics:

- Software Requirements
- Preparing the Siebel Environment
- Preparing the Server Environment for Adapter Configuration
- Configuring the Data Adapter for Siebel
- Managing Siebel Metadata

The Data Adapter for Siebel allows applications to access Siebel data sources. The adapter converts data or application requests into native Siebel statements and returns optimized answer sets to the requesting program.

Software Requirements

Verify that Java 2 SDK Version 1.2.2 or higher is installed on your environment. This can be completed by issuing the following command from a command prompt:

```
Java -version
```

If a version is not returned, contact your system administrator.

Preparing the Siebel Environment

To prepare the Siebel environment for adapter configuration, you must

- Set the Siebel Security.
- Access a Siebel Server.

Setting Security

The Data Adapter for Siebel offers two methods of user authentication:

- **Explicit.** User IDs and passwords are explicitly stated in SET CONNECTION_ATTRIBUTES commands. You can include these commands in the server global profile, edasprof.prf, for all users.
- **Password Passthru.** User ID and password received from the client application are passed to the Siebel Server for authentication. When a client connects to the server, the user ID and password are passed to Siebel for authentication and are not authenticated by the server.

Accessing a Server

Using the standard rules for deploying the Siebel Client, the server supports connections to:

- Local Siebel database servers.
- Remote Siebel database servers.

To connect to a Siebel database server, the server must be pointing to the correct Siebel Gateway Server, Siebel Enterprise Server, Siebel Object Manager, and Siebel Server on the target machine.

Once you are connected to a Siebel database server, that server may define Siebel DATABASE LINKs that can be used to access Siebel tables on other Siebel database servers.

Preparing the Server Environment for Adapter Configuration

To prepare the server environment for adapter configuration, you must

- Define environment variables.
- Add a JSCOM3 Listener.

Defining Environment Variables

The Data Adapter for Siebel requires two server environment variables. If you are configuring the Data Adapter for Siebel on Windows, see *How to Define Configuration Environment Variables for Java 2 SDK on Windows* on page 35-3. For UNIX, see *How to Define Configuration Environment Variables for Java 2 SDK on UNIX* on page 35-4.

Syntax How to Define Configuration Environment Variables for Java 2 SDK on Windows

Go to the Control Panel, double click on the system icon and click on the Advanced tab. Click the Environment Variables button and define the following *system* variables

```
CLASSPATH= jar1;jar2
PATH=J2sdk_jre_bin_client;%PATH
```

where:

`jar1`

Is equal to the full path specification of the SiebelJI_Common Java Class file, sibjicom.jar.

`jar2`

Is equal to the full path specification of the SiebelJI_enu Java Class file, sibjienu.jar.

`J2sdk_jre_bin_client`

Is equal to the full path of the Java 2 SDK JRE bin client directory.

Note: Contact your Siebel Administrator for access to these Siebel Java Class files. These files must reside on the same machine as the iWay Server. The files may be copied from a machine that has a Siebel installation.

Example Defining Configuration Environment Variables on Windows

```
CLASSPATH=D:\IBI\sibjienu.jar;D:\IBI\sibjicom.jar
PATH=D:\J2SDK1.4.0\JRE\BIN\CLIENT
```

Syntax **How to Define Configuration Environment Variables for Java 2 SDK on UNIX**

Modify your server's profile to include the following export commands needed to define the CLASSPATH and LIBPATH environment variables:

```
export CLASSPATH=jar1:jar2: $CLASSPATH
export LIBPATH=jre_bin:jre_bin_classic:$LIBPATH
```

where:

`jar1`

Is equal to the full path specification of the SiebelJI_Common Java Class file, sibjicom.jar.

`jar2`

Is equal to the full path specification of the SiebelJI_enu Java Class file, sibjienu.jar.

`jre_bin`

Is equal to the full path of the Java 2 SDK bin directory.

`jre_bin_classic`

Is equal to the full path of the Java 2 SDK bin/classic directory.

Note: Contact your Siebel Administrator for access to these Siebel Java Class files. These files must reside on the same machine as the iWay Server. The files may be copied from a machine that has a Siebel installation.

Example **Defining Configuration Environment Variables on UNIX**

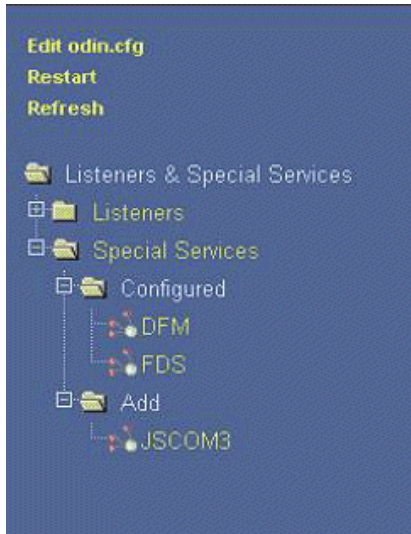
```
export CLASSPATH=/u1/ibi/sibjienu.jar:/u1/ibi/sibjicom.jar:$CLASSPATH
export LIBPATH=/usr/java131/jre/bin:/usr/java131/jre/bin/classic:$LIBPATH
```

Adding a JSCOM3 Listener

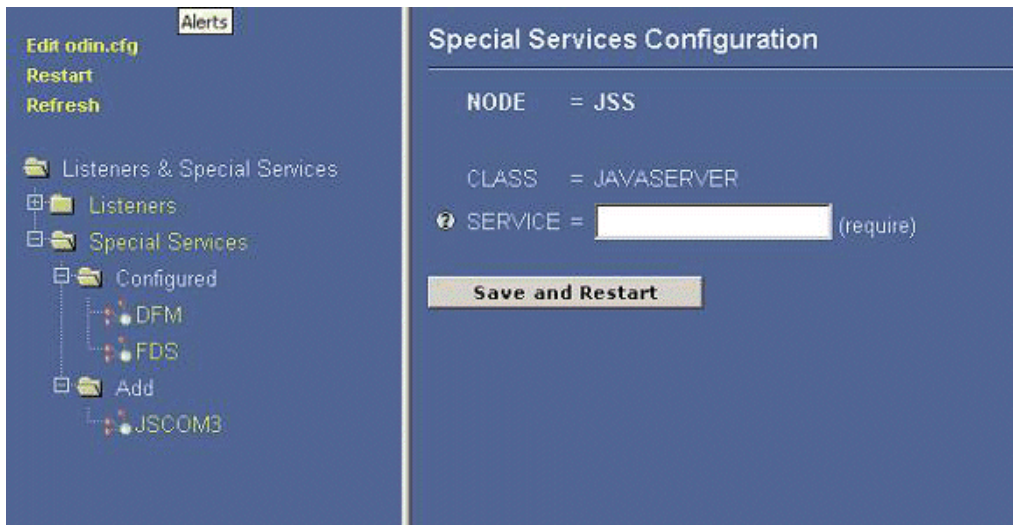
To prepare the server environment for adapter configuration, you must add a JSCOM3 Listener.

Procedure How to Add a JSCOM3 Listener

1. Start the Web Console and, in the navigation pane, click *Listeners*. After selecting *Listeners*, a new left pane appears.



2. Expand the *Add* folder from the *Special Services* folder and select *JSCOM3*. When you select this option a new right pane appears.



3. In the *Service* field, enter the next available port number for your server.
4. Click *Save and Restart*.

Configuring the Data Adapter for Siebel

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to Siebel, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Manually add the command in the global server profile (edasprof.prf) or in a user profile (user.prf).
- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the global server profile.

You can declare connections to more than one Siebel data source by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to Siebel takes place when the first query that references that connection is issued.

If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Syntax **How to Declare Connection Attributes Manually**

Explicit. The user ID and password are explicitly stated in SET CONNECTION_ATTRIBUTES commands. You can include these commands in the server global profile, edasprof.prf, for all users.

```
ENGINE SIBIN SET CONNECTION_ATTRIBUTES connection_name
/user_ID,password::'Gateway/Enterprise/Obj Mngr/Siebel'
```

Password Passthru. The user ID and password received from the client application are passed to the Siebel Server for authentication. When a client connects to the server, the user ID and password are passed to Siebel for authentication and are not authenticated by the server.

```
ENGINE SIBIN SET CONNECTION_ATTRIBUTES connection_name
:'Gateway/Enterprise/Obj Mngr/Siebel'
```

where:

SIBIN

Indicates the Data Adapter for Siebel.

connection_name

Is a logical name used to identify this particular set of connection attributes. In the Access File, it becomes the CONNECTION= attribute.

user_ID

Is the primary authorization ID by which you are known to Siebel.

password

Is the password associated with the primary authorization ID.

'Gateway/Enterprise/Obj Mngr/Siebel'

Is the Siebel connection string. This information must be supplied.

Gateway

Is the Siebel Gateway Server name.

Enterprise

Is the Siebel Enterprise Server name.

Obj Mngr

Is the Siebel Object Manager name.

Siebel

Is the Siebel Server name.

Example Declaring Connection Attributes

Explicit authentication:

```
ENGINE SIBIN SET CONNECTION_ATTRIBUTES CON1
/USER_ID,PASSWORD;:'ARIBA01/ARIBA01/SCCObjMgr/SiebelSrv'
```

Password Passthru authentication:

```
ENGINE SIBIN SET CONNECTION_ATTRIBUTES CON2
:'devportal2/siebel/EAIObjMgr/devportal2'
```

Reference Declaring Connection Attributes from the Web Console

Field	Description
Connection name	Is a logical name used to identify this particular set of connection attributes.
Gateway server	Is the Siebel Gateway Server name.
Enterprise server	Is the Siebel Enterprise Server name.
Object Manager	Is the Siebel Object Manager name.
Siebel server	Is the Siebel Server name.
Security	<p>There are two methods by which a user can be authenticated when connecting to an Siebel database server:</p> <p>Explicit. The user ID and password are explicitly specified for each connection and passed to Siebel, at connection time, for authentication.</p> <p>Password Passthru. The user ID and password received from the client application are passed to Siebel, at connection time, for authentication. This option requires that the server be started with security off.</p>
User	Is the username by which you are known to Siebel. This field only displays when the security mode of the server is non-trusted.
Password	Is the password associated with the username. This field only displays when the security mode of the server is non-trusted.

Overriding the Default Connection

Once the connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET CONNECTION_ATTRIBUTES command.

You can also include the CONNECTION attribute in the Access File of the Business Component specified in the current query. When you include a connection name in the CREATE SYNONYM command from the Web Console, the CONNECTION attribute is automatically included in the Access File. This attribute supercedes the default connection.

Syntax How to Change the Default Connection

```
ENGINE SIBIN SET DEFAULT_CONNECTION connection_name
```

where:

SIBIN

Indicates the Data Adapter for Siebel.

connection_name

Is the connection name specified in a previously issued SET CONNECTION_ATTRIBUTES command. If this connection name has not been previously declared, a FOC44221 message is issued.

Note: If you use the SET DEFAULT_CONNECTION command more than once, the connection_name specified in the last command will be the active one.

Example Selecting CON1 as the Default Connection

The following example selects the previously defined connection named CON1 as the default Siebel connection:

```
ENGINE SIBIN SET DEFAULT_CONNECTION CON1
```

Managing Siebel Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Siebel data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Business Component (BC) that is accessible from a server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

Procedure How to Create a Synonym Using the Web Console

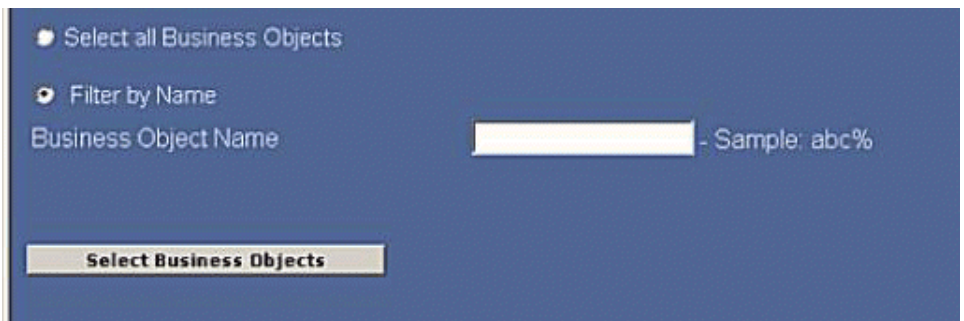
To create a synonym using the Web Console:

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Editing Metadata page opens.

Note: To create a synonym, you must have configured the adapter.

2. Expand the Add folder, then expand the adapter folder, and click a connection. The right pane displays selection options:
 - **Select All Business Objects.** Select this option button to create synonyms for all Business objects. This value is the default.
 - **Filter by Name.** Select this option button to filter the Business Objects which to create synonyms.

Selecting this option adds the following:



The screenshot shows a dark blue panel with two radio button options. The first option, 'Select all Business Objects', is selected. The second option, 'Filter by Name', is unselected. Below the 'Filter by Name' option is a text input field labeled 'Business Object Name' with a placeholder value 'Sample: abc%'. At the bottom of the panel is a button labeled 'Select Business Objects'.

Business Object Name. Enter a string for filtering the Business Objects. Inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select Business Objects which begin with the letters ABC; %ABC to select Business Objects which end with the letters ABC; %ABC% to select Business Objects which contain the letters ABC at the beginning, middle, or end.

3. Click *Select Business Objects*. All Business Objects that meet the specified criteria are displayed:

Help

Business Component selection

DBMS = SIBIN ; Conname = CON1 ; Object Name = Ac%

Select all Business Components

Filter by Name

Select Business Components

Check / Uncheck all Business Objects

Check	Business Object Name
<input checked="" type="checkbox"/>	Accept/Reject
<input checked="" type="checkbox"/>	Access Group
<input checked="" type="checkbox"/>	Account
<input checked="" type="checkbox"/>	Account - ESP
<input checked="" type="checkbox"/>	Account - Get SAP Order List
<input checked="" type="checkbox"/>	Account - Import SAP Order (Get SAP Order List)
<input checked="" type="checkbox"/>	Account - Import SAP Order (Siebel Order)
<input checked="" type="checkbox"/>	Account Category
<input checked="" type="checkbox"/>	Account Person Admin
<input checked="" type="checkbox"/>	Action

Next you need to choose the business components to use.

- **Select All Business Components.** Select this option button to create synonyms for all Business Components for every selected Business Object. This value is the default.
- **Filter by Name.** Select this option button to filter the Business Components which to create synonyms.

Selecting this option adds the following:

Help

Business Component selection

DBMS = SIBIN ; Conname = CONT ; Object Name = Ac%

Select all Business Components

Filter by Name

Business Component Name

Business Component Name. Enter a string for filtering the Business Component. Inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select Business Component which begin with the letters ABC; %ABC to select Business Component which end with the letters ABC; %ABC% to select Business Component which contain the letters ABC at the beginning, middle, or end.

4. Click *Select Business Component*. All Business Component that meet the specified criteria are displayed:

Help

Create Synonym

DBMS = SIBIN ; Conname = CONT ; Component Name = Ac%

Prefix:

Select Application Directory:

All Select

Check	Default Synonym Name	Business Object Name	Business Component Name
<input checked="" type="checkbox"/>	<u>Accept_Reject</u>	Accept/Reject	Accept/Reject
<input checked="" type="checkbox"/>	<u>Access_Group</u>	Access Group	Access Group
<input checked="" type="checkbox"/>	<u>Access_Group_Member</u>	Access Group	Access Group Member
<input checked="" type="checkbox"/>	<u>Account</u>	Account	Account
<input checked="" type="checkbox"/>	<u>Account_Get_SAP_O</u>	Account	Account - Get SAP Order List Header
<input checked="" type="checkbox"/>	<u>Account_Attachment</u>	Account	Account Attachment
<input checked="" type="checkbox"/>	<u>Account_Category</u>	Account	Account Category
<input checked="" type="checkbox"/>	<u>Account_External_Prod</u>	Account	Account External Product

5. Select which Business Object/Business Component you would like to create a synonym.
6. If you have Business Object/Business Component with identical names, assign a prefix to distinguish them. Note that the resulting synonym name cannot exceed 64 characters. If all Business Object/Business Component have unique names, leave field blank.
7. From the Select Application Directory drop-down list, select a directory. The default value is baseapp.
8. Complete your Business Object/Business Component selection:
To select all Business Object/Business Component in the list, click *All*.
To select specific Business Object/Business Component, click the corresponding check boxes.
9. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.
10. Click *Create Synonym*. Synonyms are created and added under the specified application directory.

Reference **Managing Synonyms**

In the navigation pane, click the name of the synonym to access the following options:

Option	Description
Sample Data	Retrieves up to 20 rows from the associated Business Object/Business Component.
Properties	Displays a graphic representation of the synonym and enables you to edit its metadata.
Edit Master File	Enables you to manually edit the synonym's Master File. Note: It is strongly recommended that you use the Refresh option, rather than manually editing the Master File, to update the synonym.
Edit Access File	Enables you to manually edit the synonym's Access File. Note: It is strongly recommended that you use the Refresh option, rather than manually editing the Access File, to update the synonym.
Refresh	Regenerates the synonym. Use this option if structural changes were made to the source table or view.
Drop	Deletes the synonym.

Option	Description
Copy to	Copies the synonym to another application directory. Click the target directory from the resulting list.
Move to	Moves the synonym to another application directory. Click the target directory from the resulting list.

Syntax

How to Create a Synonym Manually

```
CREATE SYNONYM appname/synonym FOR business_object/business_component
DBMS SIBIN [AT connection_name] [PARM 'VERSION {6|7}']
END
```

where:

appname

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, then this application name must not be used.

synonym

Is an alias for the Siebel Business Component (maximum 64 characters for Windows NT and UNIX server platforms).

business_object/business_component

Is the Siebel Business Object and the Siebel Business Component.

AT *connection_name*

Is the *connection_name* as previously specified in a SET CONNECTION_ATTRIBUTES command. Default connection is used if this parameter omitted. When the server creates the synonym, this *connection_name* becomes the value for CONNECTION= in the Access File.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note:

- CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.
- CREATE SYNONYM creates a Master File and Access File which represents the server metadata.

Example CREATE SYNONYM**Generated Master File account.mas**

```

FILE=SIBTST7, SUFFIX=SIBIN, $

SEGMENT=Account, SEGTYPE=S0, $
FIELD=ACCOUNT_STATUS,          ALIAS=Account Status,          USAGE=A20,
ACTUAL=A20, $
FIELD=MAIN_PHONE_NUMBER,       ALIAS=Main Phone Number,       USAGE=A20,
ACTUAL=A20, $
FIELD=NAME,                     ALIAS=,                          USAGE=A20,
ACTUAL=A20, $
FIELD=PRIMARY_EMPLOYEE_LOGIN,  ALIAS=Primary Employee Login,  USAGE=A20,
ACTUAL=A20, $
FIELD=VISIBILITY_MODE,        ALIAS=VISIBILITY_MODE,        USAGE=A15,
ACTUAL=A15,
ACCEPT=SalesRep OR Manager OR Personal OR All OR Organization OR Contact
OR Group OR Catalog OR SubOrganization,$

SEGMENT=Bill To First Name, SEGTYPE=S0, PARENT=Account, $
FIELD=BILL_TO_FIRST_NAME,      ALIAS=First Name,              USAGE=A20,
ACTUAL=A20, $
FIELD=BILL_TO_LAST_NAME,       ALIAS=Last Name,              USAGE=A20,
ACTUAL=A20, $

SEGMENT=Street Address, SEGTYPE=S0, PARENT=Account, $
FIELD=STREET_ADDRESS,          ALIAS=Street Address,          USAGE=A20,
ACTUAL=A20, $
FIELD=CITY,                     ALIAS=City,                    USAGE=A20,
ACTUAL=A20, $
FIELD=STATE,                    ALIAS=State,                   USAGE=A20,
ACTUAL=A20, $
FIELD=COUNTRY,                  ALIAS=Country,                 USAGE=A20,
ACTUAL=A20, $
FIELD=AddrNUM,                 ALIAS=ORDER,                   USAGE=I4,
ACTUAL=I2, $

```

Generated Access File account.acx

```

SEGNAME      = Account,
BS_OBJECT    = Account,
BS_COMPONENT = Account,
VERSION      = 7,
CONNECTION   = CON1, $
FLDNAME     = NAME,
FLDNAME_LONG = very_very_very_very_very_long_name_up_to_75_characters, $

```

The Master File root segment describes Siebel Business Component. All scalar fields of the Business Component are described as fields in the root segment. All Multi Value fields (MVG) of the Business Component are described as descendant segments of the root one. Name of the segment representing an MVG is derived from the Business Component Multi Value field name. All other Multi Value fields related to the same Multi Value Link are not shown in the Master File. MVG segment contains references to the appropriate fields in the linked Business Component instead. All scalar fields of the MVG are described as fields in the MVG segment. All MVG fields of an MVG become descendant segments of the segment representing this parent MVG.

Master File field aliases represent real Siebel Business Component field names. Long names (between 67 and 75 characters) are stored in the Access File. Corresponding Master File fields must have no aliases. Similarly, long segment names are stored in the Access File as well.

Special virtual field VISIBILITY_MODE is added to the root segment to enable client to set the visibility type for the Business Component. Available visibility modes are enumerated in the ACCEPT parameter for this field. Siebel versions 6 and 7 support different sets of visibility modes, what is reflected in the ACCEPT list.

The visibility modes are:

- Siebel v7: SalesRep, Manager, Personal, All, Organization, Contact, Group, Catalog, and SubOrganization.
- Siebel v6: SalesRep, Manager, Personal, All, and NoneSet.

Syntax

How to Set Visibility Mode

```
IF VISIBILITY_MODE EQ 'All'
```

This syntax tells Siebel to look at the user ID referenced in the .prf file and according to that user's ID authorization privileges, Siebel returns a view of ALL for the requested output.

If no visibility mode is set in the report, it defaults to the personal visibility mode.

Reference Access File Attributes

Keyword	Description
SEGNAME	Is the name of the corresponding segment in the Master File.
SEGNAME_LONG	Is the long (1 to 75 characters) Siebel segment name.
CONNECTION= <i>connection_name</i>	Indicates access to specified Siebel Server. Defaulted to the default connection.
BS_OBJECT	Is the Siebel Business Object name.
BS_COMPONENT	Is the Siebel Business Component name.
VERSION	Is the Siebel version number: 6 or 7. The default value is 7.
FLDNAME	Is the name of the corresponding field in the Master File.
FLDNAME_LONG	Is the long (1 to 75 characters) Siebel field name.

Data Type Support

All Siebel scalar, alphanumeric, and currency fields are treated as alpha-numeric fields. All Siebel date and integer fields are supported.

CHAPTER 36

Getting Started in Supra

Topics:

- Preparing the Supra Environment
- Configuring the Data Adapter for Supra
- Supra Overview and Mapping Considerations
- Managing Supra Metadata
- Supra Modules
- Adapter Tracing

The Data Adapter for Supra allows applications to access Supra data sources. The adapter converts application requests into native Supra statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

The following topics discuss how the Data Adapter for Supra is configured and how data is mapped to its corresponding server counterparts in the OS/390 and z/OS environment.

Preparing the Supra Environment

Prior to starting the server, it is necessary to perform the following pre-configuration procedures when preparing the environment on OS/390 and z/OS.

Procedure How to Customize the LINKEDIT JCL

To customize the LINKEDIT JCL:

1. After executing ISETUP, the JCL will be generated to LINKEDIT the supplied Data Adapter for Supra module with the Supra stub called CSVILUV. The CSVILUV module is supplied by the Supra vendor, Cincom, and usually resides in the Supra LINKLIB library.

This required step allows the Data Adapter for Supra module to communicate with the Supra Central PDM.

2. Modify the following LINKEDIT JCL to conform to site standards and submit it for execution. The JCL listed below displays the GENEFSJCL that is generated.

```
//LINK1      EXEC PGM=IEWL,PARM='XREF,LET,LIST,MAP,SIZE=1024K'  
//SYSLIB    DD  DUMMY  
//SYSPRINT  DD  SYSOUT=*  
//SYSUT1    DD  UNIT=SYSDA,SPACE=(CYL,(10,1))  
//SUPRA     DD  DSN=cincom.LINKLIB,DISP=SHR  
//SYSLMOD   DD  DSN=prefix.TOTAL.LOAD,DISP=SHR  
//MAINTAIN  DD  DSN=prefix.TOTAL.DATA,DISP=SHR  
//SYSLIN    DD  *  
            INCLUDE SUPRA(CSVILUV)  
            INCLUDE SYSLMOD(FSP000)  
            INCLUDE MAINTAIN(FSP000)  
            NAME FSP000(R)  
/*  
//
```

where:

prefix

Is the high-level qualifier for the server production libraries.

cincom.LINKLIB

Is the name of the Supra load library supplied by Cincom that contains the CSVILUV module.

prefix.TOTAL.LOAD

Is the name of the adapter load library.

`prefix.TOTAL.DATA`

Is the name of the adapter data library.

Upon successful completion of the LINKEDIT JCL, the adapter module will be ready for use with the server.

Procedure How to Customize the Server EDASTART JCL

To customize the server EDASTART JCL, add the Supra Load Libraries to STEPLIB in the server's EDASTART JCL for Supra access.

Note: The Supra DBMS has three load libraries: LINKLIB, INTERFLM, and ENVLIB. These load libraries must be concatenated to ddname STEPLIB.

Procedure How to Allocate CSIPARM

It is required that you allocate ddname CSIPARM.

This ddname must point to the data set that contains the CSIPARM definition, which in turn points to the Central PDM you are accessing. Contact your Supra DBA for the name of the file

```
//CSIPARM) DD('cincom.CSIPARM(CONNECT)'),DISP=SHR
```

where:

`cincom`

Is the high-level qualifier for the Supra libraries supplied by Cincom.

Procedure How to Set the CSISYSIN Parameters

The Data Adapter for Supra's multi-session facility provides multi-session capability for server tasks to access the Supra Physical Data Manager using the call Database facility or the Relational Data Manager (RDM). It connects with the PDM on the first access by a task. The connecting task is used solely as the connection task and is not usable for other operations. The number of task and concurrent operations supported during the session is controlled by input parameters specified in the dottedest allocated to ddname CSISYSIN.

There is no security provided by the multi-session adapter. However, the multi-session facility does provide a security exit that can be used to control user access to the database resource. This security exit is identified with the "SECURITY =" keyword described in the next section.

The parameters used for connecting the multi-session adapter to the Central PDM is allocated to ddname CSISYSIN in the server JCL as follows:

```
//CSISYSIN DD DSN=prefix.TOTAL.DATA(SUPRCNFG),DISP=SHR
```

The options available are listed below: "xx" denotes a value to be supplied by the user. Keywords must start in column 1 and each must be specified on a separate input statement. An asterisk (*) in column 1 indicates a commented line.

Keyword	Description
THREADS=xx	Is the number of concurrent requests that can be issued to the PDM. If this number is too small, a "TFUL" status may result. Refer to the <i>Supra PDM Administrator Guide</i> for more information.
TASK=xx	Is the maximum number of tasks that can be signed on to the PDM at any one time, including two tasks used for the Autostart facility and for adapter tracking. Therefore, if TASK=10, up to 8 users can be signed on. If this value is exceeded, a "CFUL" or "MFUL" status may result. The "MFUL" status is from the multi-session adapter and indicates that the task tables are full. A "CFUL" is from the PDM and indicates the same for the PDM.
SECURITY=xxxxxxx	Is the name of the user written security module that will validate access of users to the PDM data resources. If provided, it must reside in a library accessible from the multi-session adapter.
DEBUG=xxx	Is a diagnostic facility that will provide console messages for problem tracing. Refer to <i>Adapter Tracing</i> on page 36-17 for details.
END.	Is the ending statement for the parameters. The period is required.

Procedure How to Configure the Adapter Batch Autostart Facility

When using the Multi-Session Facility for Supra with the server, the first task that issues a request to the Central PDM will be used by the multi-session adapter to establish a permanent connection between the Central PDM and the server. The initial request, if executed from a server client, would tie up that client until the server was shut down. To avoid this scenario, use the Batch Autostart Facility to initialize and establish the connection between the server and Central PDM regions.

Certain parameters in the EDAPROF profile can interfere with the operation of the Batch Autostart Facility. If the service block associated with the Autostart facility (GATEKEPR) executes properly, but no connection is made to Supra, there may be a conflict between your profile and the Autostart facility. To bypass the profile, use the MSOINFO subroutine at the top of your profile to avoid executing the rest of the profile.

The following sample code uses Dialogue Manager to request information about SERVICE and branch out of the profile if appropriate:

```
-SET &REQ = 'SERVICE ' ;
-SET &ANS = '          ' ;
-TSO RUN MSOINFO,&REQ,&ANS
-IF &ANS = 'GATEKEPR' GOTO END
```

Note: If you bring up the server before the Supra PDM is up, the Autostart connection will not be established; also, if the PDM is recycled, the Autostart connection will be broken. To connect under these circumstances, use the server console to restart the service block associated with the Autostart task (SERVICE=GATEKEPR).

Procedure How to Define the GATEKEPR Service Block

The server configuration file must be modified to include an additional service block. This service block is used by the server to execute an exec at startup time which will run as a background task and establish the link between the server and the PDM. The syntax for adding the additional service block is

```
SERVICE = GATEKEPR
BEGIN
  PROGRAM = TSCOM3
  NUMBER_READY = 1
  MAXIMUM = 1
  RUNCOUNT = 5
  AUTOSTART = BATCH
PROFILE = baseapp/supprof
END
```

where:

SERVICE

Identifies and names a service. The name that you choose should be unique and from 1 to 8 characters in length. The keywords and values that follow the SERVICE keyword in each service block define the type of service to be provided. GATEKEPR is the recommended name.

PROGRAM

This parameter should always be set to TSCOM3.

NUMBER_READY

Is the number of instances of this service to be preloaded by the server. This parameter should always be set to 1.

MAXIMUM

Is the number of instances of this service allowed by the server. This parameter should always be set to 1.

RUNCOUNT

Specifies the number of attempts to connect to the central PDM. There is no default value, therefore you must specify a value to prevent the server from continually attempting to access Supra data.

PROFILE

Identifies the file containing a simple request against the Central PDM. The result of this request is the establishment of a multi-session communications link between the server and the Central PDM. Any simple request against a Supra data source can be used for establishing the link.

Note: All parameters must start in column 1 in the configuration file. Any line starting with an asterisk (*) will be considered a comment.

To summarize, the following items are required to implement the Batch Autostart Facility:

- CSISYSIN
- GATEKEPR service block in EDASERVE
- Simple request against Supra data source

Procedure How to Install the Data Adapter for Supra Security Exit (optional)

The Data Adapter for Supra has a security exit, FSPEXT, that allows the user to obtain the account name and account password from a user defined source. The account name and account password are then passed to the adapter and used for login to the Central PDM.

When a user issues a request against the Supra database, the adapter calls the FSPEXT function with standard IBM calling conventions, and passes it these parameters:

USR	Passed to function (Server Logon User ID)	08 Bytes Alpha
ACCNT	Returned from function	16 Bytes Alpha
ACCNTP	Returned from function	16 Bytes Alpha

The function returns ACCNT and ACCNTP to the adapter which uses them as parameters to the signon call it issues. The adapter calls the user exit if the password is not available from other sources (for example, the SET command or an Access File). The exit is loaded as a module using a LOAD macro from ddname USERLIB, TASKLIB or STEPLIB in that order. It is called in AMODE 31. The syntax for the call to the FSPEXT function is:

```
CALL FSPEXT(USR,ACCNT,ACCNTP)
```

Procedure How to Utilize the Security Exit

To utilize the security exit the user must:

1. Write the function using COBOL or Assembler. The function name must be FSPEXT. An example of the FSPEXT function written in Assembler follows.

```

* SAMPLE EXIT FOR Supra ADAPTER

      EJECT
FSPEXT CSECT
      USING FSPEXT,R15          TEMPORARY ADDRESSABILITY
      B      PROC              JUMP AROUND THE ID INFORMATION
* ID INFORMATION
      DC     CL8'FSPEXT '      PROG NAME
      PRINT NOGEN
PROC STM   R14,R12,12(R13)
      LR    R12,R15
      DROP R15
      USING FSPEXT,R12
      ST    R13,SAVAR+4
      MVC   8(4,R13),=A(SAVAR)
      LA   R13,SAVAR
*
*      INSERT THE CODE TO FETCH THE ACCOUNT /PASSWORD
      L     R2,4(R1)
      MVC   0(16,R2),=CL16'EXITACCOUNT '
      L     R2,8(R1)
      MVC   0(16,R2),=CL16'EXITPASS '
*
      L     R13,SAVAR+4
      LM   R14,R12,12(R13)
      BR   R14
*****
      SAVAR DS    18F
      R1   EQU   1
      R2   EQU   2
      R9   EQU   9
      R10  EQU   10
      R11  EQU   11
      R12  EQU   12
      R13  EQU   13
      R14  EQU   14
      R15  EQU   15
*
      END

```

2. LINKEDIT the object module into the 'prefix.TOTAL.LOAD' library with AMODE(31). For example

```
//LINK      EXEC PGM=IEWL,PARM='LET,NCAL,SIZE=(1024K),LIST'  
//OBJLIB    DD   DSN=objlib,DISP=SHR  
//SYSLMOD   DD   DSN=prefix.TOTAL.LOAD,DISP=SHR  
//SYSUT1    DD   UNIT=SYSDA,SPACE=(CYL,(10,1))  
//SYSPRINT  DD   SYSOUT=*  
//SYSLIN    DD   *  
            INCLUDE OBJLIB(FSPEXT)  
            ENTRY FSPEXT  
            MODE AMODE(31)  
            NAME FSPEXT(R)  
/*  
//
```

where:

objlib

Is the name of the library containing FSPEXT object code.

prefix.TOTAL.LOAD

Is the adapter load library.

3. If you LINKEDIT the object module into a load library other than 'prefix.TOTAL.LOAD', concatenate the load library from Step 2 to the STEPLIB DD in your server JCL.

Configuring the Data Adapter for Supra

Configure the adapter using the Web Console Adapter Add screen and click *Configure*.

Declaring Connection Attributes

In order to access data on the Supra Central PDM (Physical Data Manager), you must have a valid user name and password. The user name and password can be issued to the central PDM in several ways:

- Hard code the user name and password in the Access File with the "USER=" and "PASSWORD=" attributes.
- Issue the Data Adapter for Supra SET commands from any supported server profile. The syntax is:

```
ENGINE SUPRA SET USER userid
```

```
ENGINE SUPRA SET PASSWORD password
```

- Invoke the security exit. The security exit, described in *How to Install the Data Adapter for Supra Security Exit (optional)* on page 36-6, allows you to code a procedure that returns the user name and password to the Data Adapter for Supra which are used when the signon call is issued to the central PDM.

The adapter ascertains the user name and password with the following steps:

1. First, it checks the Access File.
2. If the user name and password are not coded in the Access File, the adapter checks whether the user issued the SET commands to set the user name and password.
3. If the adapter has not resolved the user name and password using the Access File or SET commands, it invokes the security exit.

Supra Overview and Mapping Considerations

This topic explains how the Data Adapter for Supra allows you to describe and report from Supra database views. You should become familiar with these topics, because most of the concepts affect the way Supra files are described to the server.

Mapping Concepts

Each Supra database can be described as a segment in a Master File. You can describe multiple views in a single Master File if the key of one view is a field in the other view. This embedded cross reference is described with the KEYFLD and IXFLD parameters in the *Access File* on page 36-13.

Supra RDM

A Supra database is a collection of one or more views. The Supra Relational Data Manager (RDM) provides access to physical fields from one or more files in a flat, two-dimensional format. A set of field values is a row. A Supra database consists of one or more views, each of which consists of one or more rows. You can define a subset of rows or reorder the columns according to your needs. This subset of data is called a user view. A column is the smallest logical unit of data a program or user can request.

After creating the view, the database administrator (DBA) defines it in the directory. The DBA can also designate one or more columns as keys to the view. The keys can be used to locate a specific set of rows.

In summary, the terms essential for understanding the data model are as follows:

View	A set of one or more rows as defined by the Database Administrator (DBA).
User view	A subset of a view which may consist of all or part of the view.
Row	A set of one or more related data items stored in computer memory.
Column	In a row, a specified area used for a particular category of data.
Value	A quantity assigned to a constant, variable, parameter, or symbol.
Key	One or more data items, the contents of which identify the type or location of a row or the ordering of data.

Keys can appear anywhere in the view. The DBA can define different types of keys:

- **Simple Unique Key.** The simplest view has one unique key. This key allows you to select and retrieve data. A unique key must have a valid, non-null value.
- **Compound Unique Key.** The DBA designates several columns as unique logical keys, and the combination of the key values is unique; that is, an "and" connection between the columns is implied. For example, to check customer orders for a certain part number, you would use a view with both customer number and part number as key values. RDM retrieves the specific customer number and part number combination if it is present.
- **Simple Non-unique Key.** This kind of key allows more than one row to contain the same value in a key column. An example is a customer file where a list of notes or comments about each customer is stored, and they are not dated or distinct. In this case, the customer number could be a simple non-unique key. When the program does its first GET using a customer number, it retrieves the first comment for this customer. A subsequent GET retrieves the second, and so on. After RDM retrieves the last comment for that customer, it will reach a boundary condition and return a NOT FOUND to the program.
- **Compound Non-unique Key.** This is an extension of the simple non-unique key. Here, more than one column is defined as a non-unique key. However, all the non-unique keys together still do not completely describe the record occurrence as unique. More than one record with the same compound non-unique key may exist.

You can access a set of rows by assigning values to the keys of the view (if there are any). The DBA determines which columns are keys and defines them on the directory. You can use the keys to locate a specific record or to perform a generic read. You can also access a set of rows sequentially by not supplying any values for the keys.

Mapping of Supra Views and Fields

To map a Supra structure to the server you must set up two files, the Master File and the Access File.

Supra concepts correspond to the following Master File elements:

- One view or user view can be defined as a single segment.
- A column from a Supra view or user view becomes a field.

The Master File should include only needed columns from the Supra view. Performance may be improved if you do not specify all the columns defined in the view. Even though it is possible to define the columns in an arbitrary order, rearranging key columns may adversely affect performance.

- For each field, the ALIAS must be defined and must be the Supra view column name.

The *Access File* on page 36-13 contains Supra DBMS specific information.

Managing Supra Metadata

This topic explains how the Data Adapter for Supra allows you to describe and report from Supra database views. You should become familiar with these topics, because most of the concepts affect the way Supra files are described to the server.

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Supra data types.

Master File

The syntax for describing a Supra database in a Master File is like that of any other Master File and uses the same keywords. A Master File can contain a File record, Segment records, and Field records. The keywords are described in the following sections.

Reference File Keywords

Keyword	Format or Value
FILENAME	Is an arbitrary name up to 8 characters in length.
SUFFIX	Is SUPRA when accessing RDM or PDM data using the Supra RDM. If RDM is not available, the SUFFIX=TOTIN is used for PDM data access. For more information, see <i>Supra PDM</i> on page 36-15.

Reference Segment Keywords

Keyword	Format or Value
SEGNAME	Is the segment name.
SEGTYPE	Is the segment type. Sn means that the first n fields are components of the key, in the order listed. This information is used only if the Access file is missing, or if no value is specified for the KEYNAME attribute of the Access File. If the two values disagree, the value specified in the Access File is retained; a message is displayed only if FSTRACE is allocated to the screen or to a file. FSTRACE is described in Appendix B.
PARENT	Is the segment name of the parent. If you describe several Supra views or user views as a hierarchical structure in one Master File, then you must define one view as the root segment and all other views as descendant segments. Each descendant segment must include the PARENT attribute to identify its parent in the hierarchy. You must also identify the shared fields for each pair of segments by specifying the KEYFLD and IXFLD attributes in the <i>Access File</i> on page 36-13.

Reference Field Keywords

Keyword	Format or Value
FIELD	Is any name; 66 characters is the maximum length.
ALIAS	Is the name of the view's column as defined to Supra. The total length must not exceed 66 characters.
USAGE	Is the usage format values are A (Alpha), I (Integer), F (Floating point), D (Decimal), or P (Packed).

Keyword	Format or Value
ACTUAL	<p>Specifies the actual Supra format (that is, the format defined by the DBA in the view definition that an application program would use for accessing the data).</p> <p>Some examples of Supra formats and their corresponding ACTUAL formats are:</p> <ul style="list-style-type: none"> • Supra B 1, 2, 4, 8 (with decimal) fields may be defined as F or D. • Supra C 1-32,767 fields may be defined as A (the maximum length of an alphanumeric field is 256 characters). • Supra F 4, 8, 16 fields may be defined as I. • Supra P 1-16 (with decimal) fields may be defined as P. <p>The ACTUAL format does not specify the number of decimal places in the number. The USAGE format defines the number of decimal places; it must be large enough to accommodate all of the digits in the number, the decimal point, and a possible minus sign. For example, if the Supra format is P5 with 2 decimal places, you can use the following specifications:</p> <p style="text-align: center;"><code>ACTUAL=P5 , USAGE=P8.2</code></p> <ul style="list-style-type: none"> • Supra Z 1-18 (with decimal) fields may be defined as Z.

Note: You can place different views into one Master File. If they have shared fields (fields with the same value in both views), then one view is described as the parent segment, and the other views are described as the child segments.

Access File

The Access File provides the information necessary for selecting the appropriate Supra database, view, and access strategy (that is, the linking of items between different tables).

A logical record in the Access file consists of a list of attributes (keyword = value pairs) separated by commas and terminated by a comma and dollar sign (,\$). The list is freeform and can span several lines. The keywords can be specified in any order.

The Access file contains two types of logical records: HEADER and SEGMENT.

Reference **HEADER Logical Record**

Keyword	Value
USER	Identifies the Supra user name. If omitted, it defaults to FOCUS.
PASSWORD	Identifies the Supra user password. If omitted, it defaults to FOCUS.

Reference **SEGMENT Logical Record**

Keyword	Value
SEGNAME	Identifies the segment name.
VIEW	Identifies the Supra view (defaults to SEGNAME).
USERVIEW	Identifies the Supra user view corresponding to the segment identified by segname (defaults to VIEW).
KEYNAME	Is a composite of at most 9 field names that constitute the key of the view; the field names are concatenated with the symbol "/", without intervening blanks. The keyword value can span more than one line. If omitted, the first "n" fields, as specified by the SEGTYPE keyword in the Master File, are assumed.
KEYFLD	Is the field name in the parent segment or a defined field that establishes the embedded cross reference. It is mandatory in all but the root segment. Its value is a composite of a maximum of 9 field names concatenated with the symbol "/", without blank spaces. The keyword value can span more than one line.
IXFLD	Is the field name in the descendant segment that establishes the embedded cross reference. It is mandatory for all but the root segment. The values of this field should match the values of the KEYFLD. The keyword value consists of a maximum of 9 field names concatenated with the symbol "/", without blanks. The keyword value can span more than one line.

Embedded Cross-Reference

If your request references fields from two segments defined in the same Master File, the two views are automatically cross-referenced. The fields that implement the cross reference are specified in the Access File with the KEYFLD and IXFLD attributes. The selection of the shared fields is transparent to you.

The following rules apply:

- As KEYFLD, you can specify:
 1. A simple field.
 2. A list of fields.
- As IXFLD, you can specify:
 1. A simple field.
 2. A list of fields.

In all cases, the length and the format of KEYFLD and IXFLD must be consistent (a list of fields may be joined to a simple field and vice versa, as long as their formats are alphanumeric).

Supra PDM

Accessing PDM data requires using SUFFIX=TOTIN in the Master File.

Supra Modules

This topic discusses the pertinent modules for the Data Adapter for Supra.

Module CFDP4001

This is a load module developed for support of the adapter's multi-session facility. This module is LINKEDITed with the name CSTEDBMI. Since this is the same name as the current Cincom single task and central region module, you must place 'prefix.TOTAL.LOAD'; the dottedst that contains this module, ahead of the LINKLIB and ENVLIB load libraries supplied by Cincom. This module is self contained and is not a composite module like the other CSTEDBMI modules. Make sure not to overlay another Cincom adapter.

Module CFDP4002

This reusable module provides the link to the adapter's multi-session facility. It must be linked as reusable so that only one copy is in the address space. The CFDP4001 module loads it when the first PDM request is processed. This module loads the multi-session adapter module (CSTEDBMI) and establishes the connection with the Central PDM. It reads the input parameters to determine the number of task and concurrent operations to be supported during the session. Any errors found will generate a console message. Depending on the error, the connection may be terminated.

Module CFDP4003

This security module provides the exit for controlling access to the PDM data resources. It is loaded by module CFDP4002 when the input parameters specify the "SECURITY=" keyword. It loads the named user module, calls the module using standard IBM calling conventions (register 1 point to the parameter list and register 13 points to the register save area), and passes a security block to the user module.

On return from the called security module, the status field is examined. The operation is accepted if the status field is "****". Any other value results in termination of the operation with a status of "SECR". The user security module may not issue any calls to the PDM. The security block passed to the module is:

Security Block Passed			
Field Name	Field Format and Values	Field Length	Starting Position
USERID	Character	8	1
ACCESS	Character (R=Read, W=Write)	1	9
DDNAME	Character	4	10
DSNNAME	Character	44	14
STATUS	Character (****=ACCESS ALLOWED)	4	58

The library containing these modules should be concatenated to the STEPLIB DD of the server JCL ahead of the LINKLIB and ENVLIB supplied by Cincom. For example,

```
//STEPLIB DD DSN=prefix.TOTAL.LOAD,DISP=SHR  
//          DD DSN=cincom.ENVLIB,DISP=SHR  
//          DD DSN=cincom.LINKLIB,DISP=SHR
```

where:

prefix

Is the high-level qualifier for the server production libraries.

prefix.TOTAL.LOAD

Is the name of the adapter load library.

cincom

Is the high-level qualifier for the Supra libraries supplied by Cincom.

Adapter Tracing

To activate the Data Adapter for Supra Tracing Facility, use the Web Console. From the Web console main page, select *Diagnostics*, and then *Traces*. Click *Enable Traces*.

The Default traces will include the Data Adapter for Supra information.

CHAPTER 37

Getting Started in Sybase ASE

Topics:

- Preparing the Sybase ASE Environment
- Configuring the Data Adapter for Sybase ASE
- Managing Sybase ASE Metadata
- Customizing the Sybase ASE Environment
- Calling a Sybase ASE Stored Procedure Using SQL Passthru

The Data Adapter for Sybase ASE allows applications to access Sybase ASE data sources. The adapter converts application requests into native Sybase ASE statements and returns optimized answer sets to the requesting application.

Preparing the Sybase ASE Environment

In order to use the Data Adapter for Sybase, you must set Sybase and platform-specific environment variables prior to starting the server. Check with your system administrator to see if these values have already been set for you.

Identifying the Location of the Interfaces File

If you are using Sybase inherent distributed access, you must set the SYBASE environment variable to identify the directory where the *interfaces* file resides. The Windows NT equivalent of the UNIX *interfaces* file is the *sql.ini* file.

The *interfaces* file is required for both local and remote access. If you do not set the SYBASE environment variable, Sybase searches /etc/passwd for the login directory of the user named Sybase and accesses the *interfaces* file in that directory.

Note that the UNIX PATH must include the location of the Sybase executables. For example, specify the PATH entry as follows for Sybase:

```
SYBASE=/usr/sybase
PATH=$PATH:$SYBASE/bin
export SYBASE PATH
```

You can export the SYBASE environment variable from the UNIX shell or in the UNIX profile of the user's machine that starts the server.

For information about other environment variables needed for Sybase executables and components, see the Sybase Installation and Configuration manual.

Specifying the Sybase Server Name

Use the DSQUERY environment variable to specify the Sybase Server name. The server uses this value only if:

- You do not issue the SET CONNECTION_ATTRIBUTES command in the global server profile (edasprof.prf).
- You do not specify the CONNECTION= attribute in the Access File.

Procedure How to Specify the Sybase Server Name

```
DSQUERY=server
export DSQUERY
```

where:

```
server
```

Is the name of the Sybase database server.

Accessing a Remote Sybase Server

Using the standard rules for deploying the Sybase Client, the server supports connections to:

- Local Sybase servers.
- Remote Sybase servers. To connect to a remote Sybase server, the *interfaces* file (*sql.ini* on Windows) must contain an entry pointing to the target machine and the listening process must be running on the target machine.

New entries in the *interfaces* file may be made using the Sybase tool DSEDIT. Entries may also be made automatically using the sybinstall facility. For details, see the Sybase documentation.

XA Support

Read/write applications accessing Sybase ASE data sources are able to perform transactions managed in XA-compliant mode.

To activate the XA Transaction Management feature, the server has to be configured in Transaction Coordination Mode, using the Web console configuration functions. Using Transaction Coordination Mode guarantees the integrity of data modification on all of the involved DBMSs and protects part of the data modifications from being committed on one DBMS and terminated on another.

For complete documentation on XA compliance, see the XA Support appendix.

Configuring the Data Adapter for Sybase ASE

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to an Sybase ASE database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Manually add the command in the global server profile (edasprof.prf) or in a user profile (user.prf).
- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the global server profile. For more information, see *Declaring Sybase ASE Connection Attributes From the Web Console*.

You can declare connections to more than one Sybase ASE database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to Sybase ASE Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Syntax

How to Declare Connection Attributes Manually

Explicit authentication. The user ID and password are explicitly specified for each connection and passed to Sybase ASE, at connection time, for authentication. The syntax is

```
ENGINE [SQLSYB] SET CONNECTION_ATTRIBUTES [server]/user_ID,password
```

Password passthru authentication. The user ID and password are explicitly specified for each connection and passed to Sybase ASE, at connection time, for authentication. This option requires that the server be started with security off. The syntax is

```
ENGINE [SQLSYB] SET CONNECTION_ATTRIBUTES [server]/
```

where:

SQLSYB

Indicates the Data Adapter for Sybase ASE. You can omit this value if you previously issued the SET SQLENGINE command.

server

Is the name of the Sybase ASE server you wish to access.

user_ID

Is the primary authorization ID by which you are known to Sybase ASE.

password

Is the password associated with the primary authorization ID.

Example Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to the Sybase ASE database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLSYB SET CONNECTION_ATTRIBUTES SAMPLESERVER/MYUSER,PASS
```

The following SET CONNECTION_ATTRIBUTES command connects to the Sybase ASE database server named SAMPLESERVER using Password Passthru authentication:

```
ENGINE SQLSYB SET CONNECTION_ATTRIBUTES SAMPLESERVER/
```

Reference Declaring Sybase ASE Connection Attributes From the Web Console

Attribute	Description
Server	Is the name of the Sybase server to access. It must match an entry in the Sybase interfaces file.
Security	There are two methods by which a user can be authenticated when connecting to Sybase: Explicit. The user ID and password are explicitly specified for each connection and passed to Sybase, at connection time, for authentication. Password Passthru. The user ID and password received from the client application are passed to Sybase, at connection time, for authentication. This option requires that the server be started with security off.
User	Is the primary authorization ID by which the user is known to Sybase.
Password	Is the password associated with the primary authorization ID. The password is stored in encrypted form.

Overriding the Default Connection

Once connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax How to Change the Default Connection

```
ENGINE [SQLSYB] SET DEFAULT_CONNECTION [connection]
```

where:

SQLSYB

Indicates the Data Adapter for Sybase ASE. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the service name defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following error message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following error message is issued: FOC1671, Command out of sequence.

Example Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the Sybase ASE database server named SAMPLENAME as the default Sybase ASE database server:

```
ENGINE SQLSYB SET DEFAULT_CONNECTION SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax **How to Control the Connection Scope**

```
ENGINE [SQLSYB] SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

SQLSYB

Indicates the Data Adapter for Sybase ASE. You can omit this value if you previously issued the SET SQLENGINE command.

FIN

Disconnects automatically only after the session has been terminated. This value is the default.

COMMIT

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Sybase ASE Metadata

When the server accesses a data source, it needs to know how to interpret the data that it finds. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Sybase ASE data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Sybase ASE table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click Metadata. The Editing Metadata page opens.

To create a synonym, you must have configured the adapter. See *Configuring the Data Adapter for Sybase ASE* on page 37-3 for more information.

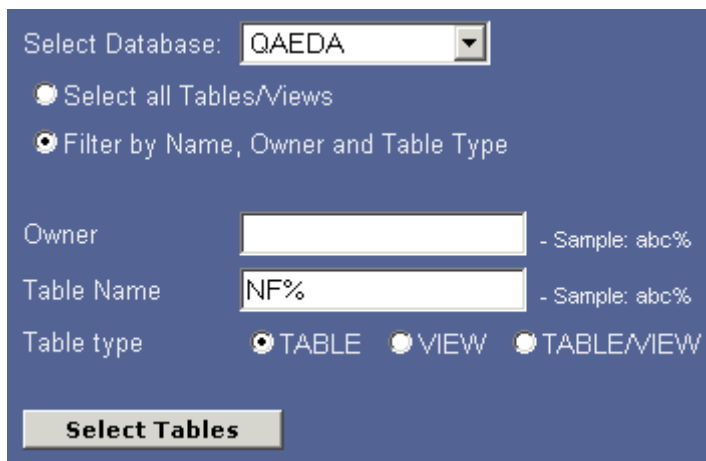
2. Expand the *Add* folder, expand the adapter folder, and then click a connection. The right pane displays table and view selection options:

Select Database. Select the database that contains the tables or views for which you want to create a synonym.

Select All Tables/Views. Select this radio button to create synonyms for all tables and views. This value is the default.

Filter by Name, Owner and Table Type. Select this radio button to filter the tables or views for which to create synonyms.

Selecting this option adds the following:



The screenshot shows a blue-themed web console interface for selecting tables and views. At the top, there is a 'Select Database:' label followed by a dropdown menu containing 'QAEDA'. Below this are two radio buttons: 'Select all Tables/Views' (which is unselected) and 'Filter by Name, Owner and Table Type' (which is selected). Under the selected radio button, there are three input fields: 'Owner' with a text box and a sample '- Sample: abc%', 'Table Name' with a text box containing 'NF%' and a sample '- Sample: abc%', and 'Table type' with three radio buttons: 'TABLE' (selected), 'VIEW' (unselected), and 'TABLE/VIEW' (unselected). At the bottom of the form is a button labeled 'Select Tables'.

Owner. Enter a string for filtering the owners' IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owners' IDs begin with the letters ABC; %ABC to select tables or views whose owners' IDs end with the letters ABC; %ABC% to select tables or views whose owners' IDs contain the letters ABC at the beginning, middle, or end.

Table Name. Enter a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

Table Type. Select one of the following options: Tables (this is the default), Views, Tables/Views.

3. Click *Select Tables*. All tables that meet the specified criteria are displayed:

Prefix: SET CONVERSION LONGCHAR:

Select Application Directory:

All Select

Check	Default Synonym Name	Owner/Schema	Table Name	Type
<input type="checkbox"/>	<input type="text" value="NF29001"/>	EDAQA	NF29001	TABLE
<input type="checkbox"/>	<input type="text" value="NF29002"/>	EDAQA	NF29002	TABLE
<input type="checkbox"/>	<input type="text" value="NF29003"/>	EDAQA	NF29003	TABLE
<input type="checkbox"/>	<input type="text" value="NF29004"/>	EDAQA	NF29004	TABLE
<input type="checkbox"/>	<input type="text" value="NF29005"/>	EDAQA	NF29005	TABLE
<input type="checkbox"/>	<input type="text" value="NF29006"/>	EDAQA	NF29006	TABLE
<input type="checkbox"/>	<input type="text" value="NF29007"/>	EDAQA	NF29007	TABLE
<input type="checkbox"/>	<input type="text" value="NF29008"/>	EDAQA	NF29008	TABLE
<input type="checkbox"/>	<input type="text" value="NF29009"/>	EDAQA	NF29009	TABLE
<input type="checkbox"/>	<input type="text" value="NF29012"/>	EDAQA	NF29012	TABLE
<input type="checkbox"/>	<input type="text" value="NF29013"/>	EDAQA	NF29013	TABLE

4. If you have tables with identical table names, assign a prefix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.
If all tables and views have unique names, leave field blank.
5. From the SET CONVERSION LONGCHAR drop-down list, select: Text, Alpha, or BLOB. The default value is Text. See *Controlling the Mapping of Large Character Data Types* on page 37-15 for more information.

6. From the Select Application Directory drop-down list, select a directory. The default value is baseapp.
7. Complete your table or view selection:
 To select all tables or views in the list, click **All**.
 To select specific tables or views, click the corresponding check boxes.
8. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.
9. Click *Create Synonym*. Synonyms are created and added under the specified application directory.

Reference **Managing Synonyms**

In the navigation pane, click the name of the synonym to access the following options:

Sample Data	Retrieves up to 20 rows from the associated data source.
Properties	Displays a graphic representation of the synonym and enables you to edit its metadata.
Edit Master File	Enables you to manually edit the synonym's Master File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Master File, to update the synonym.
Edit Access File	Enables you to manually edit the synonym's Access File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Access File, to update the synonym.
Refresh	Regenerates the synonym. Use this option if structural changes were made to the data source.
Drop	Deletes the synonym.
Copy to	Copies the synonym to another application directory. Click the target directory from the resulting list.
Move to	Moves the synonym to another application directory. Click the target directory from the resulting list.

Syntax **How to Create a Synonym Manually**

```
CREATE SYNONYM app/synonym FOR table_view [AT connection] DBMS SQLSYB
[NOCOLS]
END
```

where:

app

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, then this application name must not be used.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema) as follows:

```
[owner.] table
```

SQLSYB

Indicates the Data Adapter for Sybase ASE.

AT *connection*

Is the service name as previously specified in a SET CONNECTION_ATTRIBUTES command. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

This parameter is optional. If specified, the CONNECTION attribute is added to the Access File.

NOCOLS

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.nf29004 DBMS SQLSYB AT connsyb NOCOLS
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION ,SUFFIX=SQLSYB,$
SEGNAME=SEG1_4 ,SEGTYPE=S0 , $
FIELD=DIVISION4 ,DIVISION4 ,I9 ,I4 ,MISSING=OFF , $
FIELD=DIVISION_NA4 ,DIVISION_NA4 ,A25 ,A25 ,MISSING=ON , $
FIELD=DIVISION_HE4 ,DIVISION_HE4 ,I9 ,I4 ,MISSING=ON , $
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4 ,TABLENAME=EDAQA.nf29004 ,
CONNECTION=connsyb ,KEYS=1 , WRITE=YES , $
```

Reference Access File Keywords

Keyword	Description
TABLENAME	Identifies the Sybase table. The value assigned to this attribute can include the name of the owner (also known as schema) as follows: TABLENAME=[owner.]table
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=server Absence of the CONNECTION attribute indicates access to the default database server.

Accessing Different Databases on the Same Sybase Server

When the server connects to Sybase, it uses a primary authorization ID. This ID is associated with a default database on the server. To access tables in another database, you must specify a fully qualified name as follows:

```
database.owner.table
```

This fully qualified name can be used in the SQL request. It *must* be used in the CREATE SYNONYM command for a table in a database other than the default associated with the connected primary authorization ID. If you are allowing access to multiple databases on the same Sybase Server, the user ID specified in the SET CONNECTION_ATTRIBUTES command must have authority to access the database as well as the tables within the database. This can be achieved by using the Sybase stored procedure sp_adduser and the SQL GRANT statement. For more information, see the Sybase SQL Reference manual.

Data Type Support

The following table lists how the server maps Sybase data types. Note that you can:

- Control the mapping of large character data types.
- Change the mapping of variable-length data types.
- Change the precision and scale of numeric columns.

Be sure to review those options carefully as they affect the mapping of data types.

Sybase Data Type	Data Type		Remarks
	Usage	Actual	
CHAR (<i>n</i>)	<i>An</i>	<i>An</i>	<i>n</i> is an integer between 1 and 255
NCHAR (<i>n</i>)	<i>An</i>	<i>An</i>	<i>n</i> is an integer between 1 and 255
VARCHAR (<i>n</i>)	<i>An</i>	<i>An</i>	<i>n</i> is an integer between 1 and 255
NVARCHAR (<i>n</i>)	<i>An</i>	<i>An</i>	<i>n</i> is an integer between 1 and 255
UNICHAR			Unsupported
UNIVARCHAR			Unsupported
TEXT	A32767	A32767	
BINARY (<i>n</i>)	<i>Am</i>	<i>Am</i>	<i>n</i> is an integer between 1 and 255 <i>m</i> = 2 * <i>n</i>
VARBINARY (<i>n</i>)	<i>Am</i>	<i>Am</i>	<i>n</i> is an integer between 1 and 255 <i>m</i> = 2 * <i>n</i>
DATETIME	HYYMDs	HYYMDs	range: 1/1/1753 to 12/31/9999
SMALLDATETIME	HYYMDI	HYYMDI	range: 1/1/1900 to 6/6/2079
TIMESTAMP	A16	A16	Supported as Read-only
INT	I11	I4	range: -2 ³¹ to 2 ³¹ -1
SMALLINT	I6	I4	range: -2 ¹⁵ to 2 ¹⁵ -1
TINYINT	I6	I4	range: 0 to 255
BIT	I11	I4	"True" or "False"

Sybase Data Type	Data Type		Remarks
	Usage	Actual	
DECIMAL (p, s)	$Pn.m$	Pk	<p>p is an integer between 1 and 38 s is an integer between 0 and p</p> <p>If s is 0 and p is between 1 and 31, $n = p + 1$ If s is 0 and p is between 32 and 38, $n = 32$</p> <p>If s is greater than 0 and p is between 1 and 31, $n = p + 2$ and $m = s$ If s is greater than 0 and p is between 32 and 38, $n = 33$ and $m = 31$</p> <p>If p is between 1 and 31, $k = (p / 2) + 1$ If p is between 32 and 38, $k = 16$</p> <p>Note: If the column is nullable, p is greater than or equal to 8</p>
NUMERIC (p, s)	P33.31	P16	<p>p is an integer between 1 and 38 s is an integer between 0 and p</p>
MONEY	D20.2	D8	range: -2^{63} to $2^{63}-1$
SMALLMONEY	I11	I4	range: -214,748.3647 to 214,748.3648
FLOAT	D20.2	D8	
REAL	D20.2	D8	
IMAGE	BLOB	BLOB	<p>length: $2^{31}-1$ Supported through iWay API.</p>

Controlling the Mapping of Large Character Data Types

The SET parameter CONVERSION LONGCHAR controls the mapping of supported Sybase ASE data types listed below. By default, the server maps these data types as alphanumeric (A). The server data type A supports a maximum of 4096 characters for TABLE/MODIFY and 32768 characters for API applications.

The following table lists data type mappings based on the value of LONGCHAR:

Sybase Data Type	Remarks	LONGCHAR ALPHA or BLOB		LONGCHAR TEXT	
		Usage	Actual	Usage	Actual
TEXT		A32767	A32767	TX50	TX
BINARY (<i>n</i>)	<i>n</i> is an integer between 1 and 255 $m = 2 * n$	<i>Am</i>	<i>Am</i>	TX50	TX
VARBINARY (<i>n</i>)	<i>n</i> is an integer between 1 and 255 $m = 2 * n$	<i>Am</i>	<i>Am</i>	TX50	TX

Syntax

How to Control the Mapping of Large Character Data Types

```
ENGINE [SQLSYB] SET CONVERSION LONGCHAR {ALPHA|TEXT|BLOB}
```

where:

SQLSYB

Indicates the Data Adapter for Sybase ASE. You can omit this value if you previously issued the SET SQLENGINE command.

ALPHA

Maps the Sybase ASE data types TEXT, BINARY, and VARBINARY as alphanumeric (A). This value is the default.

TEXT

Maps the Sybase ASE data types TEXT, BINARY, and VARBINARY as text (TX). Use this value for WebFOCUS applications.

BLOB

For UNIX, Windows, OpenVMS, and OS/400, is identical to ALPHA. That is, it maps the Sybase ASE data types TEXT, BINARY, and VARBINARY as alphanumeric (A).

For OS/390 and z/OS, maps the Sybase ASE data types TEXT, BINARY, and VARBINARY as binary large object (BLOB).

Controlling the Mapping of Variable-Length Data Types

The SET parameter VARCHAR controls the mapping of the Sybase ASE data types VARCHAR, VARCHAR2, and NVARCHAR2. By default, the server maps these data types as alphanumeric (A).

The following table lists data type mappings based on the value of VARCHAR:

Sybase Data Type	Remarks	VARCHAR ON		VARCHAR OFF	
		Usage	Actual	Usage	Actual
VARCHAR (<i>n</i>)	<i>n</i> is an integer between 1 and 255	<i>AnV</i>	<i>AnV</i>	<i>An</i>	<i>An</i>
NVARCHAR (<i>n</i>)	<i>n</i> is an integer between 1 and 255	<i>AnV</i>	<i>AnV</i>	<i>An</i>	<i>An</i>
VARBINARY (<i>n</i>)	<i>n</i> is an integer between 1 and 255 $m = 2 * n$	<i>AmV</i>	<i>AmV</i>	<i>Am</i>	<i>Am</i>

Syntax How to Control the Mapping of Variable-Length Data Types

```
ENGINE [SQLSYB] SET VARCHAR {ON|OFF}
```

where:

SQLSYB

Indicates the Data Adapter for Sybase ASE. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Maps the Sybase ASE data types VARCHAR, VARCHAR2, and NVARCHAR2 as variable-length alphanumeric (*AnV*).

OFF

Maps the Sybase ASE data types VARCHAR, VARCHAR2, and NVARCHAR2 as alphanumeric (A). This value is the default.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Syntax

How to Override Default Precision and Scale

```
ENGINE [SQLSYB] SET CONVERSION RESET
ENGINE [SQLSYB] SET CONVERSION format RESET
ENGINE [SQLSYB] SET CONVERSION format [PRECISION pp [ss]]
ENGINE [SQLSYB] SET CONVERSION format [PRECISION MAX]
```

where:

SQLSYB

Indicates the Data Adapter for Sybase ASE. You can omit this value if you previously issued the SET SQLENGINE command.

RESET

Returns any previously specified precision and scale values to the data adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

format

Is any valid format supported by the data source. Possible values are:

INTEGER which indicates that the command applies only to INT columns.

DECIMAL which indicates that the command applies only to single precision floating point columns.

REAL which indicates that the command applies only to DECIMAL columns.

FLOAT which indicates that the command applies only to double precision floating point columns.

pp

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

ss

Is the scale. This is valid with DECIMAL and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. If the scale is not required, you must set mm to 0 (zero).

MAX

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
REAL	9
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example **Setting the Precision and Scale Attributes**

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE SQLSYB SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLSYB SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE SQLSYB SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLSYB SET CONVERSION RESET
```


Customizing the Sybase ASE Environment

The Data Adapter for Sybase ASE provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Specifying the Block Size for Array Retrieval

The Data Adapter for Sybase ASE supports array retrieval from result sets produced by executing SELECT queries or stored procedures. This technique substantially reduces network traffic and CPU utilization.

Using high values increases the efficiency of requests involving many rows, at the cost of higher virtual storage requirements. A value higher than 100 is not recommended because the increased efficiency it would provide is generally negligible.

Syntax How to Specify the Block Size for Array Retrieval

The block size for a SELECT request applies to TABLE FILE requests, MODIFY requests, MATCH requests, and DIRECT SQL SELECT statements.

```
ENGINE [SQLSYB] SET FETCHSIZE n
```

where:

SQLSYB

Indicates the Data Adapter for Sybase ASE. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is the number of rows to be retrieved at once using array retrieval techniques. Accepted values are 1 to 5000. The default is 20. If the result set contains a column that has to be processed as a CLOB or a BLOB, the fetchsize value used for that result set is 1.

Syntax How to Specify the Block Size for Inserting Rows

In combination with LOADONLY, the block size for an INSERT applies to MODIFY INCLUDE requests. INSERTSIZE is also supported for parameterized DIRECT SQL INSERT statements.

```
ENGINE [SQLSYB] SET INSERTSIZE n
```

where:

SQLSYB

Indicates the Data Adapter for Sybase ASE. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is the number of rows to be inserted using array insert techniques. Accepted values are 1 to 5000. The default is 1. If the result set contains a column that has to be processed as a BLOB, the insertsize value used for that result set is 1.

Activating NONBLOCK Mode

The Data Adapter for Sybase ASE has the ability to issue calls in NONBLOCK mode. The default behavior is BLOCK mode.

This feature allows the adapter to react to a client request to cancel a query while the adapter is waiting on engine processing. This wait state usually occurs during SQL parsing, before the first row of an answer set is ready for delivery to the adapter or while waiting for access to an object that has been locked by another application.

Syntax

How to Activate NONBLOCK Mode

```
ENGINE [SQLSYB] SET NONBLOCK {0|n}
```

where:

SQLSYB

Indicates the Data Adapter for Sybase ASE. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is a positive numeric number. A value of 0, the default, means that the adapter will operate in BLOCK mode. A value of 1 or greater activates the NONBLOCK calling and specifies the time, in seconds, that the adapter will wait between each time it checks to see if the:

- Query has been executed.
- Client application has requested the cancellation of a query.
- Kill Session button on the Web Console is pressed.

Note: A value of 1 or 2 should be sufficient for normal operations.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Syntax **How to Obtain the Number of Rows Updated or Deleted**

```
ENGINE [SQLSYB] SET PASSRECS {ON|OFF}
```

where:

SQLSYB

Indicates the Data Adapter for Sybase ASE. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. This value is the default.

OFF

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Calling a Sybase ASE Stored Procedure Using SQL Passthru

Sybase stored procedures are supported via SQL Passthru. These procedures need to be developed within Sybase using the CREATE PROCEDURE command.

Syntax **How to Call a Sybase Stored Procedure**

The supported syntax to call a stored procedure is shown below. It is recommended that you use the syntax below instead of the previously supported CALLSYB syntax.

```
ENGINE SQLSYB
EX SAMPLE PARM1 , PARM2 , PARM3 . . . ;
TABLE ON TABLE PCHOLD
END
```

where:

SQLSYB

Indicates the Data Adapter for Sybase. You can omit this value if you previously issued the SET SQLENGINE command.

Example **Sybase Stored Procedure**

```
CREATE PROCEDURE SAMPLE
AS
SELECT SSN5, LAST_NAME5, FIRST_NAME5, BIRTHDATE5, SEX5 FROM EDAQA.NF29005
go
exec sp_procxmode 'PROC1', 'anymode'
go
```

CHAPTER 38

Getting Started in Sybase IQ

Topics:

- Preparing the Sybase IQ Environment
- Configuring the Data Adapter for Sybase IQ
- Managing Sybase IQ Metadata
- Customizing the Sybase IQ Environment
- Calling a Sybase IQ Stored Procedure Using SQL Passthru

The Data Adapter for Sybase IQ allows applications to access Sybase IQ data sources. The adapter converts application requests into native Sybase IQ statements and returns optimized answer sets to the requesting application.

Preparing the Sybase IQ Environment

In order to use the Data Adapter for Sybase, you must set Sybase and platform-specific environment variables prior to starting the server. Check with your system administrator to see if these values have already been set for you.

Identifying the Location of the Interfaces File

If you are using Sybase inherent distributed access, you must set the SYBASE environment variable to identify the directory where the *interfaces* file resides. The Windows NT equivalent of the UNIX *interfaces* file is the *sql.ini* file.

The *interfaces* file is required for both local and remote access. If you do not set the SYBASE environment variable, Sybase searches */etc/passwd* for the login directory of the user named Sybase and accesses the *interfaces* file in that directory.

Note that the UNIX PATH must include the location of the Sybase executables. For example, specify the PATH entry as follows for Sybase:

```
SYBASE=/usr/sybase
PATH=$PATH:$SYBASE/bin
export SYBASE PATH
```

You can export the SYBASE environment variable from the UNIX shell or in the UNIX profile of the user's machine that starts the server.

For information about other environment variables needed for Sybase executables and components, see the Sybase Installation and Configuration manual.

Specifying the Sybase Server Name

Use the DSQUERY environment variable to specify the Sybase Server name. The server uses this value only if:

- You do not issue the SET CONNECTION_ATTRIBUTES command in the global server profile (edasprof.prf).
- You do not specify the CONNECTION= attribute in the Access File.

Procedure How to Specify the Sybase Server Name

```
DSQUERY=server
export DSQUERY
```

where:

```
server
```

Is the name of the Sybase database server.

Accessing a Remote Sybase Server

Using the standard rules for deploying the Sybase Client, the server supports connections to:

- Local Sybase servers.
- Remote Sybase servers. To connect to a remote Sybase server, the *interfaces* file (*sql.ini* on Windows) must contain an entry pointing to the target machine and the listening process must be running on the target machine.

New entries in the *interfaces* file may be made using the Sybase tool DSEDIT. Entries may also be made automatically using the sybinstall facility. For details, see the Sybase documentation.

Configuring the Data Adapter for Sybase IQ

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to an Sybase IQ database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Manually add the command in the global server profile (edasprof.prf) or in a user profile (user.prf).

- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the global server profile. For more information, see *Declaring Sybase IQ Connection Attributes From the Web Console*.

You can declare connections to more than one Sybase IQ database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to Sybase IQ Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Syntax

How to Declare Connection Attributes Manually

Explicit authentication. The user ID and password are explicitly specified for each connection and passed to Sybase IQ, at connection time, for authentication. The syntax is

```
ENGINE [SQLSYB] SET CONNECTION_ATTRIBUTES [server]/user_ID,password
```

Password passthru authentication. The user ID and password are explicitly specified for each connection and passed to Sybase IQ, at connection time, for authentication. This option requires that the server be started with security off. The syntax is

```
ENGINE [SQLSYB] SET CONNECTION_ATTRIBUTES [server]/
```

where:

SQLSYB

Indicates the Data Adapter for Sybase IQ. You can omit this value if you previously issued the SET SQLENGINE command.

server

Is the name of the Sybase IQ server you wish to access.

user_ID

Is the primary authorization ID by which you are known to Sybase IQ.

password

Is the password associated with the primary authorization ID.

Example Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to the Sybase IQ database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLSYB SET CONNECTION_ATTRIBUTES SAMPLESERVER/MYUSER,PASS
```

The following SET CONNECTION_ATTRIBUTES command connects to the Sybase IQ database server named SAMPLESERVER using Password Passthru authentication:

```
ENGINE SQLSYB SET CONNECTION_ATTRIBUTES SAMPLESERVER/
```

Reference Declaring Sybase IQ Connection Attributes From the Web Console

Attribute	Description
Server	Is the name of the Sybase server to access. It must match an entry in the Sybase interfaces file.
Security	There are two methods by which a user can be authenticated when connecting to Sybase: Explicit. The user ID and password are explicitly specified for each connection and passed to Sybase, at connection time, for authentication. Password Passthru. The user ID and password received from the client application are passed to Sybase, at connection time, for authentication. This option requires that the server be started with security off.
User	Is the primary authorization ID by which the user is known to Sybase.
Password	Is the password associated with the primary authorization ID. The password is stored in encrypted form.

Overriding the Default Connection

Once connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax How to Change the Default Connection

```
ENGINE [SQLSYB] SET DEFAULT_CONNECTION [connection]
```

where:

`SQLSYB`

Indicates the Data Adapter for Sybase IQ. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the service name defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following error message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following error message is issued: FOC1671, Command out of sequence.

Example **Selecting the Default Connection**

The following SET DEFAULT_CONNECTION command selects the Sybase IQ database server named SAMPLENAME as the default Sybase IQ database server:

```
ENGINE SQLSYB SET DEFAULT_CONNECTION SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax **How to Control the Connection Scope**

```
ENGINE [SQLSYB] SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

`SQLSYB`

Indicates the Data Adapter for Sybase IQ. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. This value is the default.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Sybase IQ Metadata

When the server accesses a data source, it needs to know how to interpret the data that it finds. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Sybase IQ data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Sybase IQ table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Editing Metadata page opens.

To create a synonym, you must have configured the adapter. See *Configuring the Data Adapter for Sybase IQ* for more information.

2. Expand the *Add* folder, expand the adapter folder, and then click a connection. The right pane displays table and view selection options:

Select All Tables/Views. Select this radio button to create synonyms for all tables and views. This value is the default.

Filter by Name, Owner and Table Type. Select this radio button to filter the tables or views for which to create synonyms.

Selecting this option adds the following:

Select all Tables/Views

Filter by Name, Owner and Table Type

Owner - Sample: abc%

Table name - Sample: abc%

Table type TABLE VIEW TABLE/VIEW

Select Tables

Owner. Enter a string for filtering the owners' IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owners' IDs begin with the letters ABC; %ABC to select tables or views whose owners' IDs end with the letters ABC; %ABC% to select tables or views whose owners' IDs contain the letters ABC at the beginning, middle, or end.

Table Name. Enter a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

Table Type. Select one of the following options: Tables (this is the default), Views, Tables/Views.

3. Click *Select Tables*. All tables that meet the specified criteria are displayed:

Prefix: SET CONVERSION LONGCHAR: TEXT (for Web Focus)

Select Application Directory:

All Select

Check	Default Synonym Name	Owner/Schema	Table Name	Type
<input type="checkbox"/>	<input type="text" value="NF29001"/>	EDAQA	NF29001	TABLE
<input type="checkbox"/>	<input type="text" value="NF29002"/>	EDAQA	NF29002	TABLE
<input type="checkbox"/>	<input type="text" value="NF29003"/>	EDAQA	NF29003	TABLE
<input type="checkbox"/>	<input type="text" value="NF29004"/>	EDAQA	NF29004	TABLE
<input type="checkbox"/>	<input type="text" value="NF29005"/>	EDAQA	NF29005	TABLE
<input type="checkbox"/>	<input type="text" value="NF29006"/>	EDAQA	NF29006	TABLE
<input type="checkbox"/>	<input type="text" value="NF29007"/>	EDAQA	NF29007	TABLE
<input type="checkbox"/>	<input type="text" value="NF29008"/>	EDAQA	NF29008	TABLE
<input type="checkbox"/>	<input type="text" value="NF29009"/>	EDAQA	NF29009	TABLE
<input type="checkbox"/>	<input type="text" value="NF29012"/>	EDAQA	NF29012	TABLE
<input type="checkbox"/>	<input type="text" value="NF29013"/>	EDAQA	NF29013	TABLE

4. If you have tables with identical table names, assign a prefix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave field blank.

5. From the SET CONVERSION LONGCHAR drop-down list, select: Text, Alpha, or BLOB. The default value is Text. See *Controlling the Mapping of Large Character Data Types* for more information.
6. From the Select Application Directory drop-down list, select a directory. The default value is baseapp.
7. Complete your table or view selection:
To select all tables or views in the list, click **All**.
To select specific tables or views, click the corresponding check boxes.
8. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.

9. Click *Create Synonym*. Synonyms are created and added under the specified application directory.

Reference **Managing Synonyms**

In the navigation pane, click the name of the synonym to access the following options:

Sample Data	Retrieves up to 20 rows from the associated data source.
Properties	Displays a graphic representation of the synonym and enables you to edit its metadata.
Edit Master File	Enables you to manually edit the synonym's Master File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Master File, to update the synonym.
Edit Access File	Enables you to manually edit the synonym's Access File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Access File, to update the synonym.
Refresh	Regenerates the synonym. Use this option if structural changes were made to the data source.
Drop	Deletes the synonym.
Copy to	Copies the synonym to another application directory. Click the target directory from the resulting list.
Move to	Moves the synonym to another application directory. Click the target directory from the resulting list.

Syntax **How to Create a Synonym Manually**

```
CREATE SYNONYM app/synonym FOR table_view DBMS SQLSYB [AT connection]
[NOCOLS]
[AT '']
END
```

where:

app

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, then this application name must not be used.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema) and the database link name as follows:

```
[owner.] table [@databaselink]
```

SQLSYB

Indicates the Data Adapter for Sybase IQ.

AT *connection*

Is the service name as previously specified in a SET CONNECTION_ATTRIBUTES command. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

This parameter is optional. If specified, the CONNECTION attribute is added to the Access File.

AT ' '

Adds CONNECTION=' ' in the Access File. This indicates a connection to the local Sybase database server.

NOCOLS

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.nf29004 DBMS SQLSYB AT connsyb NOCOLS
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION ,SUFFIX=SQLSYB,$
SEGNAME=SEG1_4 ,SEGTYPE=S0 , $
FIELD=DIVISION4 ,DIVISION4 ,I9 ,I4 ,MISSING=OFF , $
FIELD=DIVISION_NA4 ,DIVISION_NA4 ,A25 ,A25 ,MISSING=ON , $
FIELD=DIVISION_HE4 ,DIVISION_HE4 ,I9 ,I4 ,MISSING=ON , $
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4 ,TABLENAME=EDAQA.nf29004 ,
CONNECTION=connsyb ,KEYS=1 , WRITE=YES , $
```

Reference Access File Keywords

Keyword	Description
TABLENAME	Identifies the Sybase table. The value assigned to this attribute can include the name of the owner (also known as schema) as follows: TABLENAME=[owner.]table
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=server CONNECTION='' indicates access to the local Sybase database server. Absence of the CONNECTION attribute indicates access to the default database server.

Accessing Different Databases on the Same Sybase Server

When the server connects to Sybase, it uses a primary authorization ID. This ID is associated with a default database on the server. To access tables in another database, you must specify a fully qualified name as follows:

```
database.owner.table
```

This fully qualified name can be used in the SQL request. It *must* be used in the CREATE SYNONYM command for a table in a database other than the default associated with the connected primary authorization ID. If you are allowing access to multiple databases on the same Sybase Server, the user ID specified in the SET CONNECTION_ATTRIBUTES command must have authority to access the database as well as the tables within the database. This can be achieved by using the Sybase stored procedure sp_adduser and the SQL GRANT statement. For more information, see the Sybase SQL Reference manual.

Data Type Support

The following table lists how the server maps Sybase data types. Note that you can:

- Control the mapping of large character data types.
- Change the mapping of variable-length data types.
- Change the precision and scale of numeric columns.

Be sure to review those options carefully as they affect the mapping of data types.

Sybase Data Type	Data Type		Remarks
	Usage	Actual	
CHAR (<i>n</i>)	<i>An</i>	<i>An</i>	<i>n</i> is an integer between 1 and 255
VARCHAR (<i>n</i>)	<i>An</i>	<i>An</i>	<i>n</i> is an integer between 1 and 255
BINARY (<i>n</i>)	<i>Am</i>	<i>Am</i>	<i>n</i> is an integer between 1 and 255 <i>m</i> = 2 * <i>n</i>
VARBINARY (<i>n</i>)	<i>Am</i>	<i>Am</i>	<i>n</i> is an integer between 1 and 255 <i>m</i> = 2 * <i>n</i>
DATE	HYYMDs	HYYMDs	
TIME	HYYMDs	HYYMDs	
TIMESTAMP	A16	A16	range: 0001-01-01 00:00:00.000000 to 9999-12-31 23:59:59.999999 Supported as Read-only.
INTEGER	I11	I4	range: -2^{31} to $2^{31}-1$
UNSIGNED INT	P11	P8	range: 0 to $2^{32}-1$
BIGINT	P21	P11	range: 2^{63} to $2^{63}-1$
UNSIGNED BIGINT	P22	P11	range: 0 to $2^{64}-1$
SMALLINT	I6	I4	range: -2^{15} to $2^{15}-1$
TINYINT	I6	I4	range: 0 to 255
BIT	I11	I4	"True" or "False"

Sybase Data Type	Data Type		Remarks
	Usage	Actual	
DECIMAL (<i>p, s</i>)	P33.3	P16	<i>p</i> is an integer between 1 and 126 <i>s</i> is an integer between 0 and <i>p</i>
NUMERIC (<i>p, s</i>)	P33.31	P16	<i>p</i> is an integer between 1 and 126 <i>s</i> is an integer between 0 and <i>p</i>
FLOAT	D20.2	D8	range: 1.175494351e-38 thru 3.40282366e+38
REAL	D20.2	D8	range: 1.175494351e-38 thru 3.40282366e+38
DOUBLE	D20.2	D8	range: 2.2250738585072014e-308 thru 1.797693134862315708e+308

Note: User-defined data types are not supported. For specifications of Sybase IQ data types, see the Sybase IQ Administrator's manual.

Controlling the Mapping of Large Character Data Types

The SET parameter CONVERSION LONGCHAR controls the mapping of supported Sybase IQ data types listed below. By default, the server maps these data types as alphanumeric (A). The server data type A supports a maximum of 4096 characters for TABLE/MODIFY and 32768 characters for API applications.

The following table lists data type mappings based on the value of LONGCHAR:

Sybase Data Type	Remarks	LONGCHAR ALPHA or BLOB		LONGCHAR TEXT	
		Usage	Actual	Usage	Actual
BINARY (<i>n</i>)	<i>n</i> is an integer between 1 and 255 $m = 2 * n$	<i>Am</i>	<i>Am</i>	TX50	TX
VARBINARY (<i>n</i>)	<i>n</i> is an integer between 1 and 255 $m = 2 * n$	<i>Am</i>	<i>Am</i>	TX50	TX

Syntax **How to Control the Mapping of Large Character Data Types**

```
ENGINE [SQLSYB] SET CONVERSION LONGCHAR {ALPHA|TEXT|BLOB}
```

where:

SQLSYB

Indicates the Data Adapter for Sybase IQ. You can omit this value if you previously issued the SET SQLENGINE command.

ALPHA

Maps the Sybase IQ data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as alphanumeric (A). This value is the default.

TEXT

Maps the Sybase IQ data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as text (TX). Use this value for WebFOCUS applications.

BLOB

For UNIX, Windows, OpenVMS, and OS/400, is identical to ALPHA. That is, it maps the Sybase IQ data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as alphanumeric (A).

For OS/390 and z/OS, maps the Sybase IQ data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as binary large object (BLOB).

Controlling the Mapping of Variable-Length Data Types

The SET parameter VARCHAR controls the mapping of the Sybase IQ data types VARCHAR, VARCHAR2, and NVARCHAR2. By default, the server maps these data types as alphanumeric (A).

The following table lists data type mappings based on the value of VARCHAR:

Sybase Data Type	Remarks	VARCHAR ON		VARCHAR OFF	
		Usage	Actual	Usage	Actual
VARCHAR (<i>n</i>)	<i>n</i> is an integer between 1 and 255	<i>AnV</i>	<i>AnV</i>	<i>An</i>	<i>An</i>
VARBINARY (<i>n</i>)	<i>n</i> is an integer between 1 and 255 $m = 2 * n$	<i>AmV</i>	<i>AmV</i>	<i>Am</i>	<i>Am</i>

Syntax **How to Control the Mapping of Variable-Length Data Types**

```
ENGINE [SQLSYB] SET VARCHAR {ON|OFF}
```

where:

SQLSYB

Indicates the Data Adapter for Sybase IQ. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Maps the Sybase IQ data types VARCHAR, VARCHAR2, and NVARCHAR2 as variable-length alphanumeric (AnV).

OFF

Maps the Sybase IQ data types VARCHAR, VARCHAR2, and NVARCHAR2 as alphanumeric (A). This value is the default.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Syntax **How to Override Default Precision and Scale**

```
ENGINE [SQLSYB] SET CONVERSION RESET
```

```
ENGINE [SQLSYB] SET CONVERSION format RESET
```

```
ENGINE [SQLSYB] SET CONVERSION format [PRECISION pp [ss]]
```

```
ENGINE [SQLSYB] SET CONVERSION format [PRECISION MAX]
```

where:

SQLSYB

Indicates the Data Adapter for Sybase IQ. You can omit this value if you previously issued the SET SQLENGINE command.

RESET

Returns any previously specified precision and scale values to the data adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

format

Is any valid format supported by the data source. Possible values are:

INTEGER which indicates that the command applies only to INT columns.

DECIMAL which indicates that the command applies only to single precision floating point columns.

REAL which indicates that the command applies only to DECIMAL columns.

FLOAT which indicates that the command applies only to double precision floating point columns.

pp

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

ss

Is the scale. This is valid with DECIMAL and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. If the scale is not required, you must set mm to 0 (zero).

MAX

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
REAL	9
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example **Setting the Precision and Scale Attributes**

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE SQLSYB SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLSYB SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE SQLSYB SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLSYB SET CONVERSION RESET
```

Customizing the Sybase IQ Environment

The Data Adapter for Sybase IQ provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Activating NONBLOCK Mode

The Data Adapter for Sybase IQ has the ability to issue calls in NONBLOCK mode. The default behavior is BLOCK mode.

This feature allows the adapter to react to a client request to cancel a query while the adapter is waiting on engine processing. This wait state usually occurs during SQL parsing, before the first row of an answer set is ready for delivery to the adapter or while waiting for access to an object that has been locked by another application.

Syntax How to Activate NONBLOCK Mode

```
ENGINE [SQLSYB] SET NONBLOCK {0|n}
```

where:

SQLSYB

Indicates the Data Adapter for Sybase IQ. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is a positive numeric number. A value of 0, the default, means that the adapter will operate in BLOCK mode. A value of 1 or greater activates the NONBLOCK calling and specifies the time, in seconds, that the adapter will wait between each time it checks to see if the:

- Query has been executed.
- Client application has requested the cancellation of a query.
- Kill Session button on the Web Console is pressed.

Note: A value of 1 or 2 should be sufficient for normal operations.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Syntax **How to Obtain the Number of Rows Updated or Deleted**

```
ENGINE [SQLSYB] SET PASSRECS {ON|OFF}
```

where:

[SQLSYB](#)

Indicates the Data Adapter for Sybase IQ. You can omit this value if you previously issued the SET SQLENGINE command.

[ON](#)

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. This value is the default.

[OFF](#)

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Note that since, by definition, the successful execution of an INSERT command always affects one record, INSERT does not generate this information.

Calling a Sybase IQ Stored Procedure Using SQL Passthru

Sybase stored procedures are supported via SQL Passthru. These procedures need to be developed within Sybase using the CREATE PROCEDURE command.

Syntax How to Call a Sybase Stored Procedure

The supported syntax to call a stored procedure is shown below. It is recommended that you use the syntax below instead of the previously supported CALLSYB syntax.

```
ENGINE SQLSYB
EX SAMPLE PARM1 , PARM2 , PARM3 . . . ;
TABLE ON TABLE PCHOLD
END
```

where:

SQLSYB

Indicates the Data Adapter for Sybase. You can omit this value if you previously issued the SET SQLENGINE command.

Example Sybase Stored Procedure

```
CREATE PROCEDURE SAMPLE
AS
SELECT SSN5, LAST_NAME5, FIRST_NAME5, BIRTHDATE5, SEX5 FROM EDAQA.NF29005
go
exec sp_procxmode 'PROC1', 'anymode'
go
```

Calling a Sybase IQ Stored Procedure Using SQL Passthru

CHAPTER 39

Getting Started in Teradata

Topics:

- Preparing the Teradata Environment
- Configuring the Data Adapter for Teradata
- Managing Teradata Metadata
- Customizing the Teradata Environment

The Data Adapter for Teradata allows applications to access Teradata data sources. The adapter converts application requests into native Teradata statements and returns optimized answer sets to the requesting application.

Preparing the Teradata Environment

In order to use the Teradata Data Adapter, NCR Teradata ODBC must be installed and configured and the path to the Teradata ODBC libraries directory must be added to SYSTEM LIBRARY PATH. See the NCR Teradata® documentation for more information about requirements on ODBC installation and configuration.

If using the OS/390 and z/OS server, no pre-configuration steps are required. Use the Web Console to configure the Teradata Adapter.

Procedure How to Set Up the Environment on UNIX

You can access Teradata using the optional UNIX environment variable, ODBCINI. The ODBCINI variable points to the absolute path of the .odbc.ini file.

For example:

```
ODBCINI=/usr/odbc/.odbc.ini
export ODBCINI
```

Note: You must not use ODBCINI, if the \$HOME directory contains the .odbc.ini file.

Configuring the Data Adapter for Teradata

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to an Teradata database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Manually add the command in the global server profile (edasprof.prf) or in a user profile (user.prf).
- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the global server profile. For more information, see *Declaring Teradata Connection Attributes From the Web Console* on page 39-4.

You can declare connections to more than one Teradata database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to Teradata Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Syntax How to Declare Connection Attributes Manually

```
ENGINE [SQLDBC] SET CONNECTION_ATTRIBUTES [connection]/user_ID,password
```

where:

SQLDBC

Indicates the Data Adapter for Teradata. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is a logical name (or a data source name) used to identify this particular set of attributes.

For the Teradata ODBC Data Adapter, this is the connect descriptor to be used for the Data Source Name (DSN) node in the .odbc.ini.

For the Teradata CLI Data Adapter, this is the Teradata Director Program number (TDPn), where n is the number. This is the value of i_dbcpath in clispb.dat file.

user_ID

Is the primary authorization ID by which you are known to Teradata.

password

Is the password associated with the primary authorization ID.

Example Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to the Teradata database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLDBC SET CONNECTION_ATTRIBUTES SAMPLESERVER/MYUSER,PASS
```

Reference Declaring Teradata Connection Attributes From the Web Console

Attribute	Description
Datasource	Enter the valid Teradata DSN (data source name). There is no default DSN, a value must be entered. For ODBC-interface this DSN name should match entry in \$HOME/.odbc.ini. file. For CLI-interface the DSN name should match the TDPn. This is the value of i_dbcpath in clispb.dat file. For OS/390 and z/OS, this field is labeled SERVER and requires a valid Teradata TDP value.
User	Is the authorization by which the user is known to Teradata.
Password	Is the password associated with the authorization ID. The password is stored in encrypted form.

Note: The release numbers on the Web Console for DBMS Add and DBMS Change panels refer to the Teradata ODBC driver, not the Teradata DBMS release.

Overriding the Default Connection

Once connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax How to Change the Default Connection

```
ENGINE [SQLDBC] SET DEFAULT_CONNECTION [connection]
```

where:

SQLDBC

Indicates the Data Adapter for Teradata. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the service name defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following error message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following error message is issued: FOC1671, Command out of sequence.

Example Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the Teradata database server named SAMPLENAME as the default Teradata database server:

```
ENGINE SQLDBC SET DEFAULT_CONNECTION SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax **How to Control the Connection Scope**

```
ENGINE [SQLDBC] SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

SQLDBC

Indicates the Data Adapter for Teradata. You can omit this value if you previously issued the SET SQLENGINE command.

FIN

Disconnects automatically only after the session has been terminated. This value is the default.

COMMAND

Disconnects automatically after each request. Depending on how often the event occurs, the AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server; it is related to the operating system and the data source.

COMMIT

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Teradata Metadata

When the server accesses a data source, it needs to know how to interpret the data that it finds. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Teradata data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Teradata table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click Metadata. The Editing Metadata page opens.

To create a synonym, you must have configured the adapter. See *Configuring the Data Adapter for Teradata* on page 39-3 for more information.

2. Expand the *Add* folder, expand the adapter folder, and then click a connection. The right pane displays table and view selection options:

Select All Tables/Views. Select this radio button to create synonyms for all tables and views. This value is the default.

Filter by Name, Owner and Table Type. Select this radio button to filter the tables or views for which to create synonyms.

Selecting this option adds the following:

The screenshot shows a dark blue panel with the following elements:

- Two radio buttons:
 - Select all Tables/Views
 - Filter by Name, Owner and Table Type
- Three input fields:
 - Owner: - Sample: abc%
 - Table name: - Sample: abc%
 - Table type: TABLE VIEW TABLE/VIEW
- A button at the bottom: **Select Tables**

Owner. Enter a string for filtering the owners' IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owners' IDs begin with the letters ABC; %ABC to select tables or views whose owners' IDs end with the letters ABC; %ABC% to select tables or views whose owners' IDs contain the letters ABC at the beginning, middle, or end.

Table Name. Enter a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

Table Type. Select one of the following options: Tables (this is the default), Views, Tables/Views.

3. Click *Select Tables*. All tables that meet the specified criteria are displayed:

The screenshot shows a web-based interface for managing metadata. At the top, there is a 'Prefix' input field, a 'SET CONVERSION LONGCHAR' dropdown menu set to 'TEXT (for Web Focus)', and a 'Select Application Directory' dropdown menu set to 'baseapp'. Below these are radio buttons for 'All' and 'Select' (which is selected), and a 'Create Synonym' button. The main part of the interface is a table with the following columns: 'Check', 'Default Synonym Name', 'Owner/Schema', 'Table Name', and 'Type'. The table contains 13 rows of data, all with 'TABLE' as the type and 'EDAQA' as the owner/schema.

Check	Default Synonym Name	Owner/Schema	Table Name	Type
<input type="checkbox"/>	NF29001	EDAQA	NF29001	TABLE
<input type="checkbox"/>	NF29002	EDAQA	NF29002	TABLE
<input type="checkbox"/>	NF29003	EDAQA	NF29003	TABLE
<input type="checkbox"/>	NF29004	EDAQA	NF29004	TABLE
<input type="checkbox"/>	NF29005	EDAQA	NF29005	TABLE
<input type="checkbox"/>	NF29006	EDAQA	NF29006	TABLE
<input type="checkbox"/>	NF29007	EDAQA	NF29007	TABLE
<input type="checkbox"/>	NF29008	EDAQA	NF29008	TABLE
<input type="checkbox"/>	NF29009	EDAQA	NF29009	TABLE
<input type="checkbox"/>	NF29012	EDAQA	NF29012	TABLE
<input type="checkbox"/>	NF29013	EDAQA	NF29013	TABLE

4. If you have tables with identical table names, assign a prefix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave field blank.

5. From the SET CONVERSION LONGCHAR drop-down list, select: Text, Alpha, or BLOB. The default value is Text. See *Controlling the Mapping of Large Character Data Types* on page 39-14 for more information.

6. From the Select Application Directory drop-down list, select a directory. The default value is baseapp.
7. Complete your table or view selection:
To select all tables or views in the list, click **All**.
To select specific tables or views, click the corresponding check boxes.
8. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.
9. Click *Create Synonym*. Synonyms are created and added under the specified application directory.

Reference **Managing Synonyms**

In the navigation pane, click the name of the synonym to access the following options:

Sample Data	Retrieves up to 20 rows from the associated data source.
Properties	Displays a graphic representation of the synonym and enables you to edit its metadata.
Edit Master File	Enables you to manually edit the synonym's Master File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Master File, to update the synonym.
Edit Access File	Enables you to manually edit the synonym's Access File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Access File, to update the synonym.
Refresh	Regenerates the synonym. Use this option if structural changes were made to the data source.
Drop	Deletes the synonym.
Copy to	Copies the synonym to another application directory. Click the target directory from the resulting list.
Move to	Moves the synonym to another application directory. Click the target directory from the resulting list.

Syntax **How to Create a Synonym Manually**

```
CREATE SYNONYM app/synonym FOR table_view DBMS SQLDBC [AT connection]  
[NOCOLS]  
END
```

where:

app

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, then this application name must not be used.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema) as follows:

```
[owner.] table
```

SQLDBC

Indicates the Data Adapter for Teradata.

AT *connection*

Is the name of the connection as previously specified in a SET CONNECTION_ATTRIBUTES command. When the synonym is created, this connection name becomes the value for the CONNECTION attribute in the Access File. This is applicable only for ODBC connections.

NOCOLS

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.NF29004 DBMS SQLDBC AT DSN_A NOCOLS
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION ,SUFFIX=SQLDBC,$
SEGNAME=SEG1_4 ,SEGTYPE=S0 ,$
FIELD=DIVISION4 ,DIVISION4 ,I9 ,I4 ,MISSING=OFF ,$
FIELD=DIVISION_NA4,DIVISION_NA4 ,A25 ,A25 ,MISSING=ON ,$
FIELD=DIVISION_HE4,DIVISION_HE4 ,I9 ,I4 ,MISSING=ON ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4, TABLENAME=EDAQA.NF29004,
CONNECTION=DSN_A, KEYS=1, WRITE=YES, $
```

Reference Access File Keywords

Keyword	Description
TABLENAME	Identifies the Teradata table. The value assigned to this attribute can include the name of the owner (also known as schema) as follows: <code>TABLENAME=[owner.]table</code>
CONNECTION	Indicates a previously declared connection, for ODBC only. The syntax is: <code>CONNECTION=connection</code>

Data Type Support

The following table lists how the server maps Teradata data types. Note that you can:

- Control the mapping of large character data types.
- Change the mapping of variable-length data types.
- Change the precision and scale of numeric columns.

Be sure to review those options carefully as they affect the mapping of data types.

Teradata Data Type	Data Type		Remarks
	Usage	Actual	
CHAR (<i>n</i>)	<i>An</i>	<i>An</i>	<i>n</i> is an integer between 1 and 3200
VARCHAR (<i>n</i>)	<i>An</i>	<i>An</i>	
LONG VARCHAR	<i>An</i>	<i>An</i>	Equivalent to VARCHAR (64000) Server supports up to 32767 bytes. All data exceeding 32K will be truncated.
SMALLINT	I6	I4	2 byte signed binary integer range: 2^{+15} to $2^{+15}-1$
INTEGER	I11	I4	4 byte binary integer Range: -2.147G to +2.147G
BYTEINT	I6	I4	1 byte signed binary integer range: -128 to +127
DECIMAL	P20.2	P8 P10	<i>n</i> is an integer between 1 and 18 <i>m</i> is an integer between 0 and <i>n</i> Actual=P10 for <i>n</i> 18. Also referred to as NUMERIC.
FLOAT	D20.2	D8	8 bytes range: $2 * 10^{+307}$ to $2 * 10^{+308}$ Same as REAL or DOUBLE PRECISION.

Teradata Data Type	Data Type		Remarks
	Usage	Actual	
DATE	YYMD	A8	Date/Time formats are comprised of several components. Not a true data value stored internally.
TIME	HHISsm	A12	Date/Time formats are comprised of several components. Not a true data value stored internally. The operation with different Teradata TIME formats depends on DateTimeFormat setting in the datasource part of the \$HOME/.odbc.ini file. The adapter requires DateTimeFormat=IAI. Neither the ODBC nor the CLI interface supports integer format of TIME (I). Only the ANSI format of TIME (AT) is supported.
TIMESTAMP	HYYMdm	A20	Date/Time formats are comprised of several components. Not a true data value stored internally.
INTERVAL	An	An	INTERVAL identifies a period of time in different ranges (YEAR, MONTH, DAY, HOUR, MIN, SEC). The Teradata external (client) representation of INTERVAL is always CHAR (n) where $n = p + x$. Precision p is from 1 to 4 and x is from 1 to 11 depends on range.
GRAPHIC			Not supported in current release of the server.
VARGRAPHIC			Not supported in current release of the server.

Teradata Data Type	Data Type		Remarks
	Usage	Actual	
LONG GRAPHIC			Not supported in current release of the server.
VARBYTE			Not supported in current release of the server.

Controlling the Mapping of Large Character Data Types

The SET parameter CONVERSION LONGCHAR controls the mapping of supported Teradata data types listed below. By default, the server maps these data types as alphanumeric (A). The server data type A supports a maximum of 4096 characters for TABLE/MODIFY and 32768 characters for API applications.

The following table lists data type mappings based on the value of LONGCHAR:

Teradata Data Type	Remarks	LONGCHAR ALPHA or BLOB		LONGCHAR TEXT	
		Usage	Actual	Usage	Actual
CHAR	<i>n</i> is an integer between 1 and 4000	<i>An</i>	<i>An</i>	TX50	TX
VARCHAR (<i>n</i>)	<i>n</i> is an integer between 1 and 4000	<i>An</i>	<i>An</i>	TX50	TX
LONG VARCHAR (<i>n</i>)	<i>n</i> is an integer between 1 and 4000	<i>An</i>	<i>An</i>	TX50	TX

Syntax

How to Control the Mapping of Large Character Data Types

```
ENGINE [SQLDBC] SET CONVERSION LONGCHAR {ALPHA|TEXT|BLOB}
```

where:

SQLDBC

Indicates the Data Adapter for Teradata. You can omit this value if you previously issued the SET SQLENGINE command.

ALPHA

Maps the Teradata data types CHAR, VARCHAR, and LON VARCHAR as alphanumeric (A). This value is the default.

TEXT

Maps the Teradata data types data types CHAR, VARCHAR, and LON VARCHAR as text (TX). Use this value for WebFOCUS applications.

BLOB

For UNIX, Windows, OpenVMS, and OS/400, is identical to ALPHA. That is, it maps the Teradata data types CHAR, VARCHAR, and LON VARCHAR as alphanumeric (A).

For OS/390 and z/OS, maps the Teradata data types CHAR, VARCHAR, and LON VARCHAR as binary large object (BLOB).

Controlling the Mapping of Variable-Length Data Types

The SET parameter VARCHAR controls the mapping of the Teradata data type VARCHAR. By default, the server maps this data type as alphanumeric (A).

The following table lists data type mappings based on the value of VARCHAR:

Microsoft SQL Server Data Type	Remarks	VARCHAR ON		VARCHAR OFF	
		Usage	Actual	Usage	Actual
VARCHAR (n)	n is an integer between 1 and 4000	AnV	AnV	An	An

Syntax

How to Control the Mapping of Variable-Length Data Types

```
ENGINE [SQLDBC] SET VARCHAR {ON|OFF}
```

where:

SQLDBC

Indicates the Data Adapter for Teradata. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Maps the Teradata data type VARCHAR as variable-length alphanumeric (AnV).

OFF

Maps the Teradata data type VARCHAR as alphanumeric (A). This value is the default.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Syntax

How to Override Default Precision and Scale

```
ENGINE [SQLDBC] SET CONVERSION RESET  
ENGINE [SQLDBC] SET CONVERSION format RESET  
ENGINE [SQLDBC] SET CONVERSION format [PRECISION pp [ss]]  
ENGINE [SQLDBC] SET CONVERSION format [PRECISION MAX]
```

where:

SQLDBC

Indicates the Data Adapter for Teradata. You can omit this value if you previously issued the SET SQLENGINE command.

RESET

Returns any previously specified precision and scale values to the data adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

format

Is any valid format supported by the data source. Possible values are:

INTEGER which indicates that the command applies only to INT columns.

DECIMAL which indicates that the command applies only to DECIMAL columns.

FLOAT which indicates that the command applies only to double precision floating point columns.

pp

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

ss

Is the scale. This is valid with DECIMAL and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. If the scale is not required, you must set *ss* to 0 (zero).

MAX

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example **Setting the Precision and Scale Attributes**

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE SQLDBC SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLDBC SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE SQLDBC SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLDBC SET CONVERSION RESET
```

Customizing the Teradata Environment

The Data Adapter for Teradata provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Controlling the Column Heading in a Request

You can use the SET COLNAME command to control the column heading in a request.

Syntax How to Control Column Headings

```
ENGINE [SQLDBC] SET COLNAME {NAME|TITLE}
```

where:

SQLDBC

Indicates the Data Adapter for Teradata. You can omit this parameter value if you previously issued the SET SQLENGINE command.

NAME

If you specify the NAME option, the Teradata column name will be used as the column heading.

TITLE

If you specify the TITLE option, the name you provide is used as the column heading. The TITLE option is the default.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Syntax How to Obtain the Number of Rows Updated or Deleted

```
ENGINE [SQLDBC] SET PASSRECS {ON|OFF}
```

where:

SQLDBC

Indicates the Data Adapter for Teradata. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. This value is the default.

OFF

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

CHAPTER 40

Getting Started in UniVerse

Topics:

- Preparing the UniVerse Environment
- Configuring the Data Adapter for UniVerse
- Managing UniVerse Metadata
- Customizing the UniVerse Environment

The Data Adapter for UniVerse allows applications to access UniVerse data sources. The adapter converts data or application requests into native UniVerse statements and returns optimized answer sets to the requesting program.

Preparing the UniVerse Environment

For the UniVerse Data Adapter, no environment variables need to be set prior to starting the server.

Configuring the Data Adapter for UniVerse

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to an UniVerse database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Manually add the command in the global server profile (edasprof.prf) or in a user profile (user.prf).
- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the global server profile. For more information, see *Declaring UniVerse Connection Attributes from the Web Console*.

You can declare connections to more than one UniVerse database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to UniVerse Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Syntax How to Declare Connection Attributes Manually

```
ENGINE [SQLUV] SET CONNECTION_ATTRIBUTES [connection][data  
source]/user_ID,password [;dbname]
```

where:

SQLUV

Indicates the Data Adapter for UniVerse. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is a logical name used to identify this particular set of attributes.

data source

Is the name of the UniVerse data source you wish to access.

user_ID

Is the primary authorization ID by which you are known to UniVerse.

password

Is the password associated with the primary authorization ID.

dbname

Also referred to as *schema*, is the name of the UniVerse database used for this connection. The database name, including path, must be enclosed in single quotes.

Example Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to the UniVerse database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLUV SET CONNECTION_ATTRIBUTES SAMPLESERVER/MYUSER,PASS
```

Reference Declaring UniVerse Connection Attributes From the Web Console

Attribute	Description
Connection Name	Specify a user name for the connection information. If not specified, the data source name will be used as the connection name.
Datasource	Specify a UniVerse data source name.
Schema	Specify a UniVerse schema name including the entire path to it enclosed in single quotes.
User	Enter the user name by which you are known to UniVerse.
Password	Enter the password associated with the user name.

Note: User name and password are only checked when connecting to a remote server. When connecting to a local UniVerse server they are ignored.

Overriding the Default Connection

Once connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax How to Change the Default Connection

```
ENGINE [SQLUV] SET DEFAULT_CONNECTION [connection]
```

where:

SQLUV

Indicates the Data Adapter for UniVerse. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the service name defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following error message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following error message is issued: FOC1671, Command out of sequence.

Example Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the UniVerse database server named SAMPLENAME as the default UniVerse database server:

```
ENGINE SQLUV SET DEFAULT_CONNECTION SAMPLENAME
```


Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax How to Control the Connection Scope

```
ENGINE [SQLUV] SET AUTODISCONNECT ON {FIN|COMMAND}
```

where:

SQLUV

Indicates the Data Adapter for UniVerse. You can omit this value if you previously issued the SET SQLENGINE command.

FIN

Disconnects automatically only after the session has been terminated. This value is the default.

COMMAND

Disconnects automatically after each request. Depending on how often the event occurs, the AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server; it is related to the operating system and the data source.

Managing UniVerse Metadata

When the server accesses a data source, it needs to know how to interpret the data that it finds. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the UniVerse data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each UniVerse table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Editing Metadata page opens.

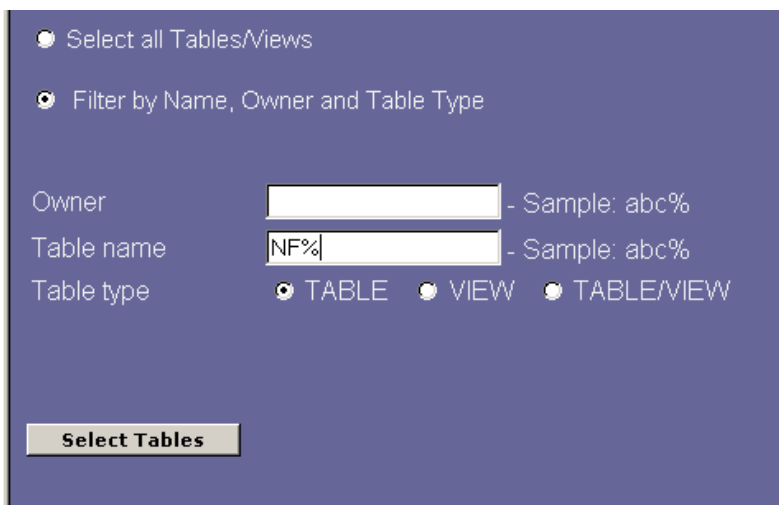
To create a synonym, you must have configured the adapter. See *Configuring the Data Adapter for UniVerse* for more information.

2. Expand the *Add* folder, expand the adapter folder, and then click a connection. The right pane displays table and view selection options:

Select All Tables/Views. Select this radio button to create synonyms for all tables and views. This value is the default.

Filter by Name, Owner and Table Type. Select this radio button to filter the tables or views for which to create synonyms.

Selecting this option adds the following:



The screenshot shows a dialog box with a dark blue background. At the top, there are two radio buttons: "Select all Tables/Views" (which is unselected) and "Filter by Name, Owner and Table Type" (which is selected). Below the radio buttons, there are three input fields: "Owner" with a text box containing a blank space and a sample text "- Sample: abc%"; "Table name" with a text box containing "NF%" and a sample text "- Sample: abc%"; and "Table type" with three radio buttons: "TABLE" (selected), "VIEW" (unselected), and "TABLE/VIEW" (unselected). At the bottom left, there is a button labeled "Select Tables".

Owner. Enter a string for filtering the owners' IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owners' IDs begin with the letters ABC; %ABC to select tables or views whose owners' IDs end with the letters ABC; %ABC% to select tables or views whose owners' IDs contain the letters ABC at the beginning, middle, or end.

Table Name. Enter a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

Table Type. Select one of the following options: Tables (this is the default), Views, Tables/Views.

3. Click *Select Tables*. All tables that meet the specified criteria are displayed:

Prefix: SET CONVERSION LONGCHAR: TEXT (for Web Focus) ▼

Select Application Directory: ▼

All Select

Check	Default Synonym Name	Owner/Schema	Table Name	Type
<input type="checkbox"/>	<input type="text" value="NF29001"/>	EDAQA	NF29001	TABLE
<input type="checkbox"/>	<input type="text" value="NF29002"/>	EDAQA	NF29002	TABLE
<input type="checkbox"/>	<input type="text" value="NF29003"/>	EDAQA	NF29003	TABLE
<input type="checkbox"/>	<input type="text" value="NF29004"/>	EDAQA	NF29004	TABLE
<input type="checkbox"/>	<input type="text" value="NF29005"/>	EDAQA	NF29005	TABLE
<input type="checkbox"/>	<input type="text" value="NF29006"/>	EDAQA	NF29006	TABLE
<input type="checkbox"/>	<input type="text" value="NF29007"/>	EDAQA	NF29007	TABLE
<input type="checkbox"/>	<input type="text" value="NF29008"/>	EDAQA	NF29008	TABLE
<input type="checkbox"/>	<input type="text" value="NF29009"/>	EDAQA	NF29009	TABLE
<input type="checkbox"/>	<input type="text" value="NF29012"/>	EDAQA	NF29012	TABLE
<input type="checkbox"/>	<input type="text" value="NF29013"/>	EDAQA	NF29013	TABLE

4. If you have tables with identical table names, assign a prefix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave field blank.

5. From the SET CONVERSION LONGCHAR drop-down list, select: Text, Alpha, or BLOB. The default value is Text.
6. From the Select Application Directory drop-down list, select a directory. The default value is baseapp.
7. Complete your table or view selection:
To select all tables or views in the list, click **All**.
To select specific tables or views, click the corresponding check boxes.
8. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.

9. Click *Create Synonym*. Synonyms are created and added under the specified application directory.

Reference **Managing Synonyms**

In the navigation pane, click the name of the synonym to access the following options:

Sample Data	Retrieves up to 20 rows from the associated data source.
Properties	Displays a graphic representation of the synonym and enables you to edit its metadata.
Edit Master File	Enables you to manually edit the synonym's Master File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Master File, to update the synonym.
Edit Access File	Enables you to manually edit the synonym's Access File. Note: It is strongly recommended that you use the Properties option, rather than manually editing the Access File, to update the synonym.
Refresh	Regenerates the synonym. Use this option if structural changes were made to the data source.
Drop	Deletes the synonym.
Copy to	Copies the synonym to another application directory. Click the target directory from the resulting list.
Move to	Moves the synonym to another application directory. Click the target directory from the resulting list.

Syntax **How to Create a Synonym Manually**

```
CREATE SYNONYM app/synonym FOR table_view DBMS SQLUV [AT connection]
[NOCOLS]
```

```
[AT ' ']
```

```
END
```

where:

app

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, then this application name must not be used.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema) and the database link name as follows:

```
[owner.] table [@databaselink]
```

SQLUV

Indicates the Data Adapter for UniVerse.

AT *connection*

Is the service name as previously specified in a SET CONNECTION_ATTRIBUTES command. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

This parameter is optional. If specified, the CONNECTION attribute is added to the Access File.

AT ' '

Adds CONNECTION=' ' in the Access File. This indicates a connection to the local Sybase database server.

NOCOLS

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

```
END
```

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.NF29004 DBMS SQLUV AT CONN_UV
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION ,SUFFIX=SQLUV,$
SEGNAME=SEG1_4 ,SEGTYPE=S0 ,$
FIELD=DIVISION4 ,DIVISION4 ,I9 ,I4 ,MISSING=OFF ,$
FIELD=DIVISION_NA4,DIVISION_NA4 ,A25 ,A25 ,MISSING=ON ,$
FIELD=DIVISION_HE4,DIVISION_HE4 ,I9 ,I4 ,MISSING=ON ,$
```

Generated Access File nf29004.acx

```
SEGNAME=NF29004 ,
TABLENAME=NF29004,
CONNECTION=CONN_UV,
KEYS=0
,$
```

Reference Access File Keywords

Keyword	Description
TABLENAME	Identifies the UniVerse tablename. The tablename may include owner- (schema-) name. For example, <code>TABLENAME=[owner.]tablename</code>
CONNECTION= <i>connection</i>	Indicates access to specified connection. When <code>CONNECTION=</code> is absent, this indicates access to the default connection.

Data Type Support

The following chart provides information about the default mapping of UniVerse data types to server data types:

UniVerse Data Type	Data Type		Remarks
	Usage	Actual	
BIT (n) in (1...2032) BIT(1) -default value	N/A	N/A	Not supported.
CHAR (n) in (1...254) CHAR(1) -default value	A(1...254)	A(1...254)	
DATE	DATE	YYMD	
DECIMAL(p,s) DEC(9)-default value	DEC(1...9)	Dec(1...31)	The server supports up to Dec(31,9) if the number is larger than 31, it will not be displayed.
DOUBLE PRECISION	D8	D20.2	
FLOAT(p)	D8	D20.2	
INTEGER	I4	I11	
NATIONAL CHARACTER	A254	A254	
NUMERIC	P8	P11	
NATIONAL CHARACTER VARYING	A32767	A32767	
REAL	D8	D20.2	
SMALLINT	I4	I6	
TIME	A8	A8	
VARBIT	N/A	N/A	Not supported.
CHAR VARYING	A32767	A32767	

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Syntax

How to Override Default Precision and Scale

```
ENGINE [SQLUV] SET CONVERSION RESET  
ENGINE [SQLUV] SET CONVERSION format RESET  
ENGINE [SQLUV] SET CONVERSION format [PRECISION pp [ss]]  
ENGINE [SQLUV] SET CONVERSION format [PRECISION MAX]
```

where:

SQLUV

Indicates the Data Adapter for UniVerse. You can omit this value if you previously issued the SET SQLENGINE command.

RESET

Returns any previously specified precision and scale values to the data adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

format

Is any valid format supported by the data source. Possible values are:

INTEGER which indicates that the command applies only to INT columns.

DECIMAL which indicates that the command applies only to DECIMAL columns.

FLOAT which indicates that the command applies only to double precision floating point columns.

pp

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

ss

Is the scale. This is valid with DECIMAL and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. If the scale is not required, you must set *ss* to 0 (zero).

MAX

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE SQLUV SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLUV SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE SQLUV SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLUV SET CONVERSION RESET
```

Customizing the UniVerse Environment

The Data Adapter for UniVerse provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Displaying Multivalued Columns in UniVerse

The OPENMODE NNF setting determines how tables with multivalued columns are treated. For the UniVerse Data Adapter to function properly, OPENMODE must be set to NNF in edasprof.prf.

Syntax **How to Set OPENMODE NNF**

```
ENGINE [SQLUV] SET OPENMODE NNF
```

where:

SQLUV

Indicates the UniVerse Data Adapter. You can omit this value if you previously issued the SET SQLENGINE command.

OPENMODE NNF

Allows the user application to see UniVerse tables the way they actually exist, including any multivalued columns.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Syntax **How to Obtain the Number of Rows Updated or Deleted**

```
ENGINE [SQLUV] SET PASSRECS {ON|OFF}
```

where:

SQLUV

Indicates the Data Adapter for UniVerse. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. This value is the default.

OFF

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Controlling Transactions

In order to perform Data Definition Language commands in UniVerse using the server, the TRANSACTION setting must be set to OFF.

Syntax **How to Issue the TRANSACTIONS Command**

```
ENGINE SQLUV SET TRANSACTIONS {OFF|ON}
```

where:

SQLUV

Indicates the UniVerse Data Adapter. You can omit this value if you previously issued the SET SQLENGINE command.

OFF

Turns transaction processing off.

ON

Turns transaction processing on. This value is the default.

CHAPTER 41

Getting Started in XML

Topics:

- Preparing the XML Environment
- Configuring the Data Adapter for XML
- Managing XML Metadata
- Using XML XFOCUS

The Data Adapter for XML allows applications to access XML data sources. The adapter converts application requests into native XML statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

Preparing the XML Environment

The Data Adapter for XML does not require any environment variables to be set up.

Configuring the Data Adapter for XML

In the Data Adapter for XML configuration screen, click *Configure* to add the Data Adapter for XML.

Associating a Master File With an XML Document

You must explicitly issue a FILEDEF command to associate a Master File with an XML document. In order to properly code the FILEDEF command, you need to determine how the XML documents are organized. If the documents are in one file and are described by the same DTD, the organization is called a COLUMN. If there are various documents in one directory and all are described by the same DTD, then the organization is called a COLLECTION.

Source	Collection Organization	Column Organization
disk	supported	supported
http	not supported	supported

Syntax

How to Associate a Master File With an XML Document

```
FILEDEF ddname source filename
```

where:

ddname

Is the logical reference name matching the desired Master File.

source

Is the location of the XML document. Possible values are [http](#) and [disk](#).

filename

For an XML Column, is the full path to the source file.

For an XML Collection, is the full path to the source directory. The directory name must be followed by the wildcard characters '*.*'.

The file name must start with 'http:/' if the XML documents are to be read using an HTTP session.

Note: HTTP is not supported for collection organization.

Example Issuing a FILEDEF Command for an XML Column

For Windows NT, when you have the ordercol.xml document under C:\, use the following FILEDEF in your profile:

```
FILEDEF ordercol disk C:\ordercol.xml
```

For UNIX, when you have the ordercol.xml document under /usera/xml/, use the following FILEDEF in your profile:

```
FILEDEF ordercol DISK /usera/xml/ordercol.xml
```

For OS/390 and Z/OS, when you have the ordercol.xml document in TEST.XML.DOC, use the following FILEDEF in your profile:

```
FILEDEF ordercol DISK //'TEST.XML.DOC'
```

For HTTP, when you have the ordercol.xml document under http://www.test.com/xmltest/, use the following FILEDEF in your profile:

```
FILEDEF order http http://www.test.com/xmltest/ordercol.xml
```

Example Issuing a FILEDEF Command for an XML Collection

For Windows NT, when you have the order1.xml and order2.xml documents under C:\ORDERS, use the following FILEDEF in your profile:

```
FILEDEF order disk C:\orders\*.*
```

For UNIX, when you have the ordercol.xml document under /usera/orders, use the following FILEDEF in your profile:

```
FILEDEF ordercol DISK //'TEST.XML.DOC(dtd)'
```

For OS/390 and Z/OS, when you have the order document in TEST.XML.DOC(dtd), use the following FILEDEF in your profile:

```
FILEDEF order DISK /usera/orders/*.*
```

Accessing XML Documents From Relational DBMS CLOB Fields

XML documents might be stored in any fields or columns in any data source. Reporting from such documents is supported by defining their structure as sub-trees attached to a parent segment which describes the original data. The CREATE SYNONYM process must be run against the data in the DBMS and against the DTD describing the XML document. The two Master Files must then be manually combined to make the DTD Master File a child of the Master File created against the DBMS. A FILEDEF is not needed in this instance.

Managing XML Metadata

When the server accesses a data source, it needs to know how to interpret the data that it finds. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the XML data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each XML data structure that is accessible from a server. Synonyms are useful because they hide the location and identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while enabling client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

For details on how to create a synonym through the Web Console, see the *Server Configuration and Operation* manual for your platform. The CREATE SYNONYM command creates a Master File and an Access File based on a given DTD. The DTD can be embedded within an XML document or be an external DTD.

Syntax How to Create a Synonym Manually

```
CREATE SYNONYM appname/synonym FOR xmlname DBMS XML AT location
```

where:

appname

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used. If your server is not APP enabled, then this application name must not be used.

synonym

Is an alias for the data source. This value must correspond to the logical name defined in the FILEDEF command.

xmlname

Is the name of the data source. The data source can be either an XML document or a DTD document.

location

Indicates the location of the data source. On Windows NT/2000 and UNIX, this may be a local full path or a URL. If it is a URL, it must start with http://.

In the case of HTTP, *location* may be a relative path to the location of the XML document mentioned in the CREATE SYNONYM command. This functionality is not available when the XML document mentioned in the CREATE SYNONYM command is a local file.

Note: The data source for the CREATE SYNONYM command may be a DTD, an XML document with embedded DTD, or an XML document which points to an external DTD. In the case of the latter, the DOCTYPE statement from the XML document:

```
<!DOCTYPE root_element_name SYSTEM "dtd_location">
```

is being parsed in order to retrieve the location of the DTD.

If *location* is a full path, the DTD will be processed from that path.

If *location* is a file name, the DTD will be assumed to be under location (in the same directory as the XML document mentioned in the CREATE SYNONYM command).

Note: CREATE SYNONYM can span more than one line, however a single element cannot span more than one line.

If there is no user-defined entity in the DTD, the Access File contains a \$ sign. If there is a user-defined entity in the DTD, then the Access File contains the following for each entity

```
Entity_name Entity_value
```

where:

```
entity_name
```

Is the entity name extracted from the DTD.

```
entity_value
```

Is the entity value extracted from the DTD.

2. Create a Master File from the DTD for the XML data in the column in the table in the DBMS. If there are two DTDs, create a Master File for each DTD.
3. Manually combine the Master Files. On each root segment for the XML Master File, add three fields: position, parent and segsuf. The POSITION keyword identifies the field containing the XML document. The PARENT field describes the original data source. The field SEGSUF defines the root segment of an XML document representing sub-tree. The total length of all fields in the Master File must not exceed the FOCUS limitation of 32k. If it does, the query will fail.

```
FILENAME=BASEAPP/ORDER, SUFFIX=XML      , $
SEGMENT=ORDER, SEGTYPE=S0, $
  FIELDNAME=ORDER, ALIAS='Order', USAGE=A1, ACTUAL=A1, $
  FIELDNAME=KEY, ALIAS='key', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=CUSTOMER, ALIAS='Customer', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=STATUS, ALIAS='Status', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=TOTALPRICE, ALIAS='TotalPrice', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=DATE, ALIAS='Date', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=PRIORITY, ALIAS='Priority', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=CLERK, ALIAS='Clerk', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=SHIPPRIORITY, ALIAS='ShipPriority', USAGE=A10, ACTUAL=A10,$
  FIELDNAME=COMMENT, ALIAS='Comment', USAGE=A10, ACTUAL=A10, $
FILENAME=BASEAPP/PORDER, SUFFIX=XML      , $
SEGMENT=PORDER, SEGTYPE=S0, $
  FIELDNAME=PORDER, ALIAS='POrder', USAGE=A1, ACTUAL=A1, $
  FIELDNAME=KEY, ALIAS='key', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=CUSTOMER, ALIAS='Customer', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=ADDRESS, ALIAS='Address', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=STATE, ALIAS='State', USAGE=A10, ACTUAL=A10, $
```

Combined Master file:

```

FILE=XMLPRO1      ,SUFFIX=SQLPRO  ,$
SEGNAME=XMLPRO1  ,SEGTYPE=S0     ,$
FIELD=FLD1 ,FLD1 ,A2      ,A2  ,MISSING=ON  ,$
FIELD=FLD2 ,FLD2 ,TX50   ,TX   ,MISSING=ON  ,$
FIELD=FLD3 ,FLD33,TX50   ,TX   ,MISSING=ON  ,$
SEGMENT=ORDER,  SEGTYPE=S0,POSITION=FLD2,PARENT=XMLPRO1,SEGSUF=XML, $
  FIELDNAME=ORDER, ALIAS='Order',  USAGE=A1, ACTUAL=A1, $
  FIELDNAME=KEY, ALIAS='key',  USAGE=A10, ACTUAL=A10, $
  FIELDNAME=CUSTOMER, ALIAS='Customer',  USAGE=A10, ACTUAL=A10, $
  FIELDNAME=STATUS, ALIAS='Status',  USAGE=A10, ACTUAL=A10, $
  FIELDNAME=TOTALPRICE, ALIAS='TotalPrice',  USAGE=A10, ACTUAL=A10, $
  FIELDNAME=DATE, ALIAS='Date',  USAGE=A10, ACTUAL=A10, $
  FIELDNAME=PRIORITY, ALIAS='Priority',  USAGE=A10, ACTUAL=A10, $
  FIELDNAME=CLERK, ALIAS='Clerk',  USAGE=A10, ACTUAL=A10, $
  FIELDNAME=SHIPPRIORITY, ALIAS='ShipPriority',  USAGE=A10, ACTUAL=A10, $
  FIELDNAME=COMMENT, ALIAS='Comment',  USAGE=A10, ACTUAL=A10, $
SEGMENT=POORDER,  SEGTYPE=S0,POSITION=FLD3,PARENT=XMLPRO1,SEGSUF=XML $
  FIELDNAME=POORDER, ALIAS='Order',  USAGE=A1, ACTUAL=A1, $
  FIELDNAME=KEY, ALIAS='key',  USAGE=A10, ACTUAL=A10, $
  FIELDNAME=CUSTOMER, ALIAS='Customer',  USAGE=A10, ACTUAL=A10, $
  FIELDNAME=ADDRESS, ALIAS='Address',  USAGE=A10, ACTUAL=A10, $
  FIELDNAME=STATE, ALIAS='State',  USAGE=A10, ACTUAL=A10, $

```

Data Type Support

The CREATE SYNONYM process uses a DTD as the source for creating the synonym. DTDs do not contain data type descriptions, therefore the actual and usage attributes of all fields in the Master File are set to ALPHA data type.

If you try to perform arithmetic operations (-, +, >, <, etc.) on fields that are numbers or dates in the XML document but are mapped as ALPHA in the Master File, you may not get the expected results since the arithmetic operations are performed on string literals. To override this problem you may modify the USAGE attribute of a field as described in the following sections.

Note: The data type defined in the ACTUAL attribute of a field must remain as ALPHA.

In order to change the default setting you must issue the SET VARCHAR command.

Syntax **How to Set the Actual and Usage Attributes**

```
ENGINE XML SET VARCHAR {ON|OFF}
```

The ACTUAL and USAGE attributes of each field in the Master File is set arbitrarily to A10. You can override this setting using the SET FIELDLENGTH command

```
ENGINE XML SET FIELDLENGTH nnn
```

where:

nnn

Is the length assigned to actual and usage attributes of all fields in the Master File during the create synonym process. The maximum length is 3000.

Conversion

The data in an XML document may reflect dates or numeric values, however, all the fields in a Master File created by the CREATE SYNONYM command are set to the ALPHA data type.

Numeric Values

In order to enable arithmetic operations on numeric fields, the data type specified in the USAGE attribute of a numeric field needs to be modified, depending on the data in the XML document, to one of the following data types: Integer (I), Double Float (D) or Decimal (P). If the data type is modified to Double Float or Decimal, use scale and precision as necessary to describe the data in the XML document.

Furthermore, it is recommended that the length of the ALPHA data type specified in the ACTUAL attribute of the numeric field be modified to reflect the maximum length of the data in the XML document.

Dates in XML

In order to enable arithmetic operations on dates, the data type specified in the USAGE attribute of a date field needs to be modified, depending on the date format used in the XML document, to one of the following data types: YYMD, MDYY or DMY.

Furthermore, the length of the ALPHA data type specified in the ACTUAL attribute of the date field needs to be modified to 10.

Note:

1. Once you have set the data type in the Master File to one of the date data types, you must make sure that in each of the XML documents from which you report the date format is the same as the one you defined in the Master File. If the data types differ, an error will result.
2. The date format in the answer set is not derived from the date format used in the XML document and is always set to YYYY/MM/DD.

Example Using Dates in XML

If in the XML document you have the following format:	Then use USAGE=
1996-01-30	YYMD
01-30-1996	MDYY
30-01-1996	DMYY

Using XML XFOCUS

This new feature enables the server to receive XML documents using an MQseries queue or from a SOAP protocol stream and then perform one of two operations, EXECUTE or ARCHIVE.

Using XML XFOCUS to Execute XML Documents

The execute operation takes an XML document, extracts a request (data retrieval or RPC), processes the request, and returns an answer set in XML format to the calling environment. The eda.dtd triggers the submission for execution. The XML document must conform to the eda.dtd. This is a sample of an XML document. In this request, an sql select from car is being requested:

```
<?xml version="1.0" ?>
<!DOCTYPE xmlfoc01 SYSTEM "eda.dtd">
  <eda>
    <request>
      <connection>
        <dsn>unused</dsn>
        <user>unused</user>
        <password>unused</password>
        <sql>
          <query>select car from car</query>
        </sql>
      </connection>
    </request>
  </eda>
```

Using XML XFOCUS to Archive XML Documents

The archive operation takes any XML document, extracts its content and, preserving the hierarchical structure, stores it in a FOCUS database. The database can then be queried from a Business Intelligence viewpoint or the XML can be extracted in its original format for further processing. The steps to configure this archival procedure are:

1. Configure an MQXML listener. To configure the listener you need to have set up three local queues under your MQseries queue manager. For example, INQUEUE, OUTQUEUE and ERRORQUEUE.

From the Web Console home page, click *Listeners*. This brings you to the Listeners configuration screen where you can add MQXML as the listener. When you click *MQXML*, you will see:

QMANGER = the mqseries queue manager.

INQUEUE = the queue to which the XML document is written.

OUTQUEUE = the queue to which the status message is written.

ERRORQUEUE = the queue to which the XML document is sent if there is a problem with the configuration and it cannot be archived.

Once the listener is configured, you will need to click the button to save and restart the server. The listener (MQXML) status should now be active.

2. Configure for the XML Adapter. Go back to the home page and either click *Data Adapters* or *Configure Adapter*. This will bring you to the Configuring Data Adapters screen. From here you choose XML as the adapter you want to add and click the CONFIGURE button. When configuration is complete, you can click the selected adapter and see a pull down menu from which you can choose Create Synonym and go the XML directory input screen. You must create a synonym for the DTD of the files you will be archiving.

When the synonym is created, click the synonym to see the ARCHIVE option under the pull down menu. Click this option and you will see that three files will be created once you click the CREATE button. These files have the same names as the DTD, but with different extensions. For example, if the DTD is called SAMPLES.DTD, you will see:

SOURCE = baseapp/sample

TARGET = baseapp/sample_arc

LOAD_PROCEDURE = baseapp/sample_load

Remember the name of the load procedure because you will need it to configure the service. You must repeat the synonym creation process for each DTD you want to archive.

3. To configure a new service, go to WORKSPACE on the Web Console home page, select CONFIGURE from the pull down menu. This will bring you to the WORKSPACE CONFIGURATION screen. Click *Services* and you will jump a new page where you will see an option for service with DEFAULT. Pull down on DEFAULT and pick the NEW SERVICES option.

New service name - The new service you are adding. It must be in uppercase and be the name of the DTD.

Maximum - Accept the default (10).

Number_ready - Accept the default (2).

Deployment - Must be pooled.

Pool_user - The user ID of the administrator of the server.

Pool_password - The password of the administrator of the server.

Max_sessions_per_agent- Accept the default (1).

Queueing - Must be on.

maxium_q - Set to 60.

queue_limit - Take default (1)

idle_session_limit=Accept the default (1).

idle_agent_limit-Accept the default (120).

profile - Enter name of load procedure from the archive screen (for example, sample_load).

cpu_limit - Accept the default (1).

memory_limit - Accept the default (1).

max_connection_per_user- Accept the default (1).

4. Once finished, click the SAVE AND RESTART button to update and restart the server. You have completed the configuration for the XML XFOCUS archive.

APPENDIX A

XA Support

Topics:

- XA Transaction Management
- Supported Interfaces
- Implementation
- Vendor Specifics

This topic describes XA Transaction Management and implementation specifics.

XA Transaction Management

An application usually defines steps performed against any given resource in terms of *transactions*.

A *transaction* is a complete unit of work in which:

- Results are either all committed or all rolled back.
- The shared resource is transformed from one valid state to another.
- Changes to shared resources do not become visible outside the transaction until the transaction commits.
- The changes of commit are able to survive system or media failure.

The server has the ability to access multiple data sources and to carry out heterogeneous joins across different data sources. The transaction that describes work done by more than one Resource Manager is called a Global or Distributed Transaction.

Most relational and some non-relational resources are maintained by their Resource Managers. The X/Open Distributed Transaction Processing (DTP) model is a software architecture that allows multiple application programs to access resources provided by multiple resource managers, and allows their work to be coordinated into global transactions.

Read/write applications accessing Oracle, DB2, MS SQL Server, and Informix data sources, as well as CICS-controlled VSAM data sets, or IMS databases, are able to perform transactions managed in XA-compliant mode. The Transaction Coordinator component uses a two-phase commit protocol for completion of customer transactions across different database management systems.

Using the XA Transaction Management feature does not require any changes in existing ODBC, JDBC, or WebFOCUS Maintain applications. To activate the XA Transaction Management feature, the server has to be configured in Transaction Coordination Mode, using the Web console configuration functions. In Transaction Coordination Mode, all requests performed against the listed XA-compliant DBMSs will be processed as XA-distributed transactions, and completed in two-phase commit mode. This guarantees the integrity of data modification on all of the involved DBMSs. It protects part of the data modifications from being committed on one DBMS and terminated on another. Since CICS-controlled VSAM and IMS are not XA-compliant DBMSs, only one of them can actually participate in the server-coordinated transaction. The integrity of data modifications in this case is secured by the protocol used by the Transaction Coordinator: first it prepares the transaction for completion on all the involved XA-compliant DBMSs. Then, if the first phase of completion is successful, it tries to complete the transaction of the non-XA-compliant phase. If this is also successful, it proceeds with the second phase of completion with the XA-compliant DBMSs. Otherwise, it rolls the transaction back.

Supported Interfaces

The supported XA-compliant interfaces are:

- DB2 (using CLI interface)
- Oracle
- MS SQL Server (using OLE DB interface)
- Informix
- Sybase ASE

Supported non-XA-compliant interfaces are:

- TP-CICS
- TP-IMS

Implementation

In order to adhere to the DTP model, the server-implemented Transaction Coordinator provides the ability to manage global transactions. Although the Transaction Coordinator is a private implementation, it fully adheres to XA protocol in the boundaries of its implementation.

- Because the Transaction Coordinator is a private implementation, it does not have a public interface. There is no mechanism for foreign applications to use the Transaction Coordinator.
- Not every Resource Manager, even XA-compliant, is supported unless explicitly stated.

The implementation specifics of the Transaction Coordinator are:

- It is invoked by a keyword in edaserve.cfg, (transaction_coordination_mode=on).
- Every unit of work initiated inside any agent will be started as a global transaction, within this agent.
- Coordination will apply to all currently supported XA-compliant interfaces.
- It cannot be turned off.
- All XA-compliant engines will participate, enabled implicitly.
- All transactions will be in the scope of global transaction, each commit/rollback will subsequently do begin transaction.
- No CREATE/DROP TABLE commands in XA mode.

Vendor Specifics

The implementation specifics of DB2 are:

- The DB2CLI.INI file must have an entry in the common section
`[COMMON]`
`DISABLEMULTITHREAD=1`
- Only the global file can be used; it must reside in \$INSTHOME.

The implementation specifics of Sybase are:

- The Sybase XA configuration file is needed.
- No bulk load/fast load functionality in XA; no error messages are produced; and insert will operate in normal mode.
- Users can create their XA configuration file and point to it in the profile

Syntax

How to Create the XA Configuration File

```
ENGINE SQLSYB SET XACONFIGFILE path
```

where:

path

Should include file name.

Index

Symbols

? SET command for Bull 4-16

Numerics

24-bit mode for Adabas 2-77

A

ACCEPT attribute for RMS 32-43

ACCESS attribute 2-49, 21-28
for Adabas 2-49
for MODEL 204 21-28

Access File keywords 13-8
for DB2 7-11
for Informix 13-8
for Microsoft SQL Server 19-8
for Nucleus 25-6
for ODBC 26-5
for Oracle 27-10
for Progress 29-7
for Rdb 30-5
for Sybase ASE 37-7
for Sybase IQ 38-7
for UniVerse 40-10

Access Files 1-6, 1-11 to 1-12, 2-29
Adabas segment attributes in 2-47
DATACOM passwords in 6-2
DATACOM security codes in 6-2
for Adabas 2-21 to 2-22, 2-29, 2-35, 2-45, 2-57
for Bull 4-10
for DATACOM 6-2, 6-12
for Enterprise Java Beans (EJB) 8-6
for Essbase 9-6
for IDMS/DB 10-31
for IDMS/SQL 11-4, 11-9
for IMS 12-10
for LRF subschema in IDMS/DB 10-59
for MODEL 204 21-21, 21-31

Access Files (*continued*)

for MQSeries 22-3
for MUMPS 23-3
for network subschema for IDMS/DB 10-54
for RMS 32-37
for SAP/R3 34-18
for Siebel 35-10
for Supra 36-13, 36-15
for XML 41-4
index declarations for IDMS/DB 10-31
J.D. Edwards 16-21
keywords in Enterprise Java Beans (EJB) 8-10
mapping descriptors in Adabas 2-54
multiple DATACOM logical files in 6-16
ordering data retrieval for Adabas 2-97
release declaration for Adabas 2-46
segment declarations for IDMS/DB 10-31
selection order for Adabas 2-82
subschemas declarations for IDMS/DB 10-31 to 10-32

Access Parameter for RMS 32-34

ACCESS=AADS attribute for Adabas 2-50

accessing XML documents 41-3

ACCOUNT attribute for MODEL 204 21-23

ACCOUNTPASS attribute for MODEL 204 21-23

ACTUAL attribute 1-9, 2-43
for Adabas 2-43
for IDMS/DB 10-22
for MODEL 204 21-19
for RMS 32-9, 32-15, 32-44
for XML 41-10

Adabas Data Adapter 2-7 to 2-8, 2-11
24-bit mode 2-77
ACCESS attribute 2-49
Access File 2-21 to 2-22, 2-27, 2-35, 2-45
access types 2-94

Adabas Data Adapter (*continued*)

- ACCESS=ADBS attribute 2-50
- ACTUAL attribute 2-43
- ALIAS attribute 2-41
- ALL prefix 2-82
- application requests 2-7
- CALLTYPE attribute 2-50
- comments in Master and Access Files 2-29
- complex FIND calls 2-103, 2-107
- components 2-12
- configuring 2-9
- configuring using OS/390 and z/OS parameters 2-9
- configuring using the Web Console 2-9
- connection attributes 2-10
- CREATE SYNONYM 2-32
- creating synonyms 2-22
- creating synonyms manually 2-22
- cross-referenced files 2-70
- customizing 2-72
- Data Definition Module (DDM) 2-14
- data formats 2-41
- data retrieval 2-14
- database numbers 2-76
- DBNO attribute 2-50
- default passwords 2-10 to 2-11
- descendant periodic (PE) groups 2-105
- descendant records 2-105
- descriptors 2-19, 2-54, 2-57
- direct calls 2-94
- dynamic database numbers 2-76
- entry segment retrieval 2-95
- FETCH attribute 2-53, 2-75
- FETCHSIZE attribute 2-53, 2-75
- field definition tables (FDT) 2-15
- FIELDNAME attribute 2-41
- field-oriented record retrieval 2-97
- file attributes in Master Files 2-36 to 2-37
- file navigation 2-78
- FILENAME attribute 2-36 to 2-37
- FILENO attribute 2-50
- FIND call 2-84

Adabas Data Adapter (*continued*)

- FIND navigation 2-102
- fixed-length records 2-20
- generic parameters 2-9
- GFBID (Global Format Buffer ID) 2-29, 2-32
- Global Format Buffer ID (GFBID) 2-29, 2-32
- GROUP attribute 2-45, 2-55, 2-70
- group fields 2-87
- implementing embedded JOINS 2-59, 2-61
- INDEX attribute 2-45
- INDEX=I attribute 2-54
- Internal Sequence Number (ISN) 2-30 to 2-31
- Internal Sequence Number (ISN) parameter 2-29
- inverted lists 2-14
- ISN (Internal Sequence Number) 2-29
- IXFLD attribute 2-59
- JOIN command 2-85
- joining files in 2-85
- KEYFLD attribute 2-59
- mapping descriptors 2-54
- Master File 2-21 to 2-22, 2-26, 2-35
- metadata 2-21
- missing non-unique segments 2-79
- MU (multi-value) fields 2-39
- Multifetch option 2-74
- multi-field JOINS 2-85
- Multi-Programming Module (MPM) 2-13
- multi-value (MU) fields 2-39, 2-62, 2-65, 2-67 to 2-68, 2-105
- non-descriptor fields 2-84
- null-suppression 2-17, 2-86
- OCCURS attribute 2-40, 2-65, 2-69
- OPEN and CLOSE calls for report requests 2-46
- OPEN attribute 2-47
- optimization on group fields 2-87
- ORDER field 2-69, 2-72
- overriding default passwords 2-10
- PARENT attribute 2-40
- partial fields in superdescriptors 2-58
- PASS attribute 2-53
- PE (periodic element) groups 2-39

Adabas Data Adapter (*continued*)

- periodic element (PE) groups 2-39, 2-62, 2-65, 2-67 to 2-68
- platform specific functionality 2-71
- Predict Dictionary 2-14
- predict files 2-23
- Prefetch option 2-74
- preparing the environment 2-8
- preparing the environment on OpenVMS 2-8
- preparing the environment on UNIX 2-8
- preparing the environment on Windows NT/2000 2-8
- Read Descriptor Value (L9) direct calls 2-104
- Read Logical calls 2-98, 2-106
- Read Logical navigation 2-97
- Read Physical calls 2-96, 2-98
- READLIMIT keyword 2-83
- read-only mode 2-94
- record retrieval 2-78
- RECORDLIMIT keyword 2-83
- RELEASE attribute 2-47
- release declaration 2-46
- repeating fields 2-62
- reporting considerations 2-78
- root segments in Master Files 2-38
- segment attributes in Access Files 2-47
- segment attributes in Master Files 2-37
- segment retrieval 2-78
- SEGNAM attribute 2-49
- SEGNAME attribute 2-38
- SEGTYPE attribute 2-39
- selection criteria 2-82
- selection order in Access Files 2-82
- SEQFIELD attribute 2-53
- server file structure 2-18
- SET ALL=OFF option 2-80
- SET ALL=ON option 2-81
- SET ALL=PASS option 2-81
- SET PASSWORD command 2-10 to 2-11
- setting new synonyms 2-72 to 2-73
- simple FIND calls 2-102, 2-106
- specifying null-suppression 2-59

Adabas Data Adapter (*continued*)

- specifying subdescriptors 2-58
- standard field formats 2-16
- subdescriptors 2-57
- subtrees 2-78
- SUFFIX attribute 2-36 to 2-37
- superdescriptors 2-54, 2-57 to 2-58, 2-73, 2-87
- unique segments 2-79
- USAGE and ACTUAL values 2-55
- USAGE attribute 2-43
- variable-length records 2-21, 2-62

adapter configuration 2-9, 26-2

- for Adabas 2-9
- for Allbase 3-2
- for Bull 4-2, 4-4
- for DATACOM 6-2
- for DB2 7-3
- for Enterprise Java Beans (EJB) 8-2, 8-5
- for Essbase 9-2
- for IDMS/DB 10-2
- for IDMS/SQL 11-2
- for IMS 12-9
- for Informix 13-4
- for Ingress II 14-2
- for Interplex 15-2
- for Lawson 17-4
- for Microsoft Access 18-2
- for Microsoft SQL Server 19-3
- for Microsoft SQL Server Analytical Engine 20-3
- for MODEL 204 21-2
- for MQSeries 22-2
- for MUMPS 23-2
- for My SQL 24-2
- for Nucleus 25-3
- for ODBC 26-2
- for Oracle 27-5
- for PeopleSoft 28-42
- for Rdb 30-2
- for Red Brick 31-2
- for Supra 36-8
- for Sybase ASE 37-3
- for Sybase IQ 38-3

Index

- adapter configuration (*continued*)
 - for Teradata 39-3
 - for UniVerse 40-2
 - for XML 41-2, 41-12
- adapter customization 3-13
 - for Adabas 2-72
 - for Allbase 3-13
 - for DB2 7-24
 - for Essbase 9-12
 - for IDMS/DB 10-61
 - for IDMS/SQL 11-15
 - for Informix 13-2
 - for Microsoft Access 18-17
 - for Microsoft SQL Server 19-23
 - for Microsoft SQL Server Analytical Engine 20-8
 - for Nucleus 25-14
 - for ODBC 26-13
 - for Oracle 27-25
 - for Progress 29-17
 - for Red Brick 31-15
 - for Sybase IQ 38-19
 - for Teradata 39-17
 - for UniVerse 40-13
- adapter modules 36-15 to 36-16
 - CFDP4001 for Supra 36-15
 - CFDP4002 for Supra 36-16
 - CFDP4003 for Supra 36-16
- adapter paths for PeopleSoft 28-12
- administration reports for PeopleSoft 28-41
- advanced features for SAP/R3 34-33
- AIS (Automatic Index Selection) for RMS 32-37 to 32-38
- ALIAS attribute 1-9, 2-41
 - for Adabas 2-41
 - for Essbase 9-15 to 9-16
 - for IDMS/DB 10-21
 - for MODEL 204 21-4, 21-18
 - for RMS 32-5, 32-46
- aliases 7-29
- ALL prefix for Adabas 2-82
- ALL_TAB_COLUMNS system catalog table 1-9
- Allbase Data Adapter 3-1
 - application requests 3-1
 - data types 3-4
 - preparing the environment 3-2
- ANSI-compliant data sources for Informix 13-3
- APP for J.D. Edwards 16-6
- APP MAP for J.D. Edwards 16-6
- application requests 2-7, 3-1, 7-1
 - for Adabas 2-7
 - for Allbase 3-1
 - for Bull 4-1
 - for C-ISAM 5-1
 - for DATACOM 6-1
 - for DB2 7-1
 - for Enterprise Java Beans (EJB) 8-1
 - for Essbase 9-1
 - for IDMS/DB 10-1
 - for IDMS/SQL 11-1
 - for IMS 12-1
 - for Informix 13-1
 - for Ingres II 14-1
 - for Interplex 15-1
 - for Microsoft Access 18-1
 - for Microsoft SQL Server 19-1
 - for Microsoft SQL Server Analytical Engine 20-1
 - for MODEL 204 21-1
 - for MQSeries 22-1
 - for MUMPS 23-1
 - for MySQL 24-1
 - for Nucleus 25-1
 - for ODBC 26-1
 - for Oracle 27-1
 - for Progress 29-1
 - for Rdb 30-1
 - for Red Brick 31-1
 - for RMS 32-1
 - for SAP BW 33-1
 - for SAP/R3 34-1

application requests (*continued*)

- for Siebel 35-1
- for Supra 36-1
- for Sybase ASE 37-1
- for Sybase IQ 38-1
- for Teradata 39-1
- for XML 41-1
- J.D. Edwards 16-1

APT (Automatic Passthru) 1-3

archiving XML documents with XML XFOCUS 41-12

area sweep record retrieval for IDMS/DB 10-87

array retrieval 19-23

- for Microsoft SQL Server 19-23
- for ODBC 26-13
- for Oracle 27-25
- for Progress 29-17
- for Sybase ASE 37-19

ASF (Automatic System Facility) for IDMS/DB 10-14

attributes 1-21

authentication 13-4

- for Informix 13-4
- for Microsoft SQL Server Data Adapter 19-3
- for MODEL 204 21-2
- for Oracle 27-5
- for PeopleSoft 28-9, 28-21
- for SAP BW 33-6
- for Sybase ASE 37-3
- for Sybase IQ 38-3
- for UniVerse 40-2

AUTODISCONNECT command 13-4

- for Bull 4-9
- for Informix 13-4
- for Microsoft SQL Server 19-3
- for Nucleus 25-6
- for Red Brick 31-2
- for Teradata 39-6

Automatic Index Selection (AIS) for RMS 32-37 to 32-38

Automatic Passthru (APT) 1-3

Automatic System Facility (ASF) for IDMS/DB 10-14

B

BAPI support for SAP/R3 34-33

BEA WebLogic Application Server for Enterprise Java Beans (EJB) 8-2, 8-4

BEx Analyzer for SAP BW 33-8 to 33-9

bill-of-materials structures for IDMS/DB 10-10

block size 27-25

- for Microsoft SQL Server 19-23
- for Oracle 27-25
- for Sybase ASE 37-19

buffer capacity for MODEL 204 21-33

Bull Data Adapter 4-1 to 4-2

- ? SET command 4-16
- Access File 4-10
- application requests 4-1
- AUTODISCONNECT command 4-9
- configuring 4-2, 4-4
- configuring using the Web Console 4-4 to 4-5
- connection attributes 4-7
- connection profile 4-8
- creating synonyms 4-10
- creating synonyms manually 4-12
- creating synonyms using the Web Console 4-10
- DROP SYNONYM command 4-13
- FOCPATH command 4-15 to 4-16
- functional limitations 4-3
- GCOS7 environment 4-2
- GCOS8 environment 4-2 to 4-3
- Master File 4-10
- metadata 4-9
- open services 4-2
- operations considerations 4-2
- preparing the environment 4-2
- server settings 4-2
- SET CONNECTION_ATTRIBUTES command 4-7
- SET FTPSERVER command 4-14

Bull Data Adapter (*continued*)
 setting parameters with SET FTPSERVER 4-15
 TIMEOUT command 4-15
 TN FOCUS 4-1

Bull RTE (Run Time Environment) 4-2, 4-7
 connecting to 4-7

Business Components for Siebel 35-10

Business Objects for Siebel 35-10

C

CALC fields for IDMS/DB 10-13, 10-84, 10-89

CALLTYPE attribute for Adabas 2-50

Central PDM for Supra 36-4

Central Version (CV) mode for IDMS/DB 10-33

CHECK FILE command for IDMS/DB 10-63

C-ISAM Data Adapter 5-1 to 5-2
 application requests 5-1
 environment variables 5-2
 file locations 5-2
 Micro Focus installation directory 5-2
 preparing the environment 5-2
 specifying file locations 5-2 to 5-3
 specifying flat file locations on an HFS file system 5-4
 specifying flat file locations on OS/390 5-4
 specifying flat file locations on UNIX and Windows NT 5-3
 specifying VSAM file locations 5-5

Column Information 1-18

columnar reports for SAP BW 33-24

columns
 documenting 1-22 to 1-23

commit processing for RMS 32-56

common member structure for IDMS/DB 10-8

common owner structure for IDMS/DB 10-7

complex FIND calls for Adabas 2-103, 2-107

complex RMS keyed files 32-17

configuration 26-2
 for Adabas 2-9
 for Allbase 3-2
 for Bull 4-2, 4-4
 for DATACOM 6-2
 for DB2 7-3
 for Enterprise Java Beans (EJB) 8-2, 8-5
 for Essbase Data Adapter 9-2
 for IDMS/DB 10-2
 for IDMS/SQL 11-2
 for IMS 12-9
 for Informix 13-4
 for Informix Data Adapter 13-4
 for Ingress II 14-2
 for Interplex 15-2
 for Lawson 17-4
 for Microsoft Access 18-2
 for Microsoft Access Data Adapter 18-2
 for Microsoft SQL Server 19-3
 for Microsoft SQL Server Analytical Engine 20-3
 for MODEL 204 21-2
 for MQSeries 22-2
 for MUMPS 23-2
 for My SQL 24-2
 for Nucleus 25-3
 for ODBC 26-2
 for ODBC Data Adapter 26-2
 for Oracle 27-5
 for PeopleSoft 28-4, 28-15, 28-42
 for Progress 29-3
 for Rdb 30-2
 for Red Brick 31-2
 for SAP BW 33-5
 for SAP/R3 34-12
 for Siebel 35-6
 for Supra 36-8
 for Sybase ASE 37-3
 for Sybase IQ 38-3
 for Teradata Data Adapter 39-3
 for UniVerse 40-2
 for XML 41-2, 41-12

- CONNECT command for IDMS/SQL 11-2
- connection attributes 2-10, 13-4, 40-3
 - for Adabas 2-10
 - for Bull 4-7
 - for DATACOM 6-2
 - for DB2 7-3
 - for Enterprise Java Beans (EJB) 8-3 to 8-4
 - for Essbase 9-3
 - for IDMS/DB 10-2
 - for Informix 13-4
 - for Lawson 17-4 to 17-5
 - for Microsoft Access 18-2
 - for Microsoft SQL Server 19-3, 19-5
 - for Microsoft SQL Server Analytical Engine 20-3
 - for MUMPS 23-2
 - for MySQL 24-2
 - for Nucleus 25-3 to 25-4
 - for ODBC 26-2
 - for Oracle 27-5
 - for Progress 29-3
 - for Rdb 30-2
 - for Red Brick 31-2 to 31-3
 - for SAP BW 33-5 to 33-6
 - for SAP/R3 34-12 to 34-13
 - for Siebel 35-6 to 35-8
 - for Sybase ASE 37-3
 - for Sybase IQ 38-3
 - for Teradata 39-4
 - for UniVerse 40-2
- connection parameters for PeopleSoft 28-20
- connection profile for Bull 4-2 to 4-3, 4-8
- connection scope 8-5, 13-4
 - for Enterprise Java Beans (EJB) 8-5 to 8-6
 - for IDMS/SQL 11-3
 - for Informix 13-4
 - for Microsoft Access 18-2
 - for Microsoft SQL Server 19-3
 - for Nucleus 25-3, 25-5 to 25-6
 - for ODBC 26-2
 - for Oracle 27-5
- connection scope (*continued*)
 - for Progress 29-3
 - for Rdb 30-2
 - for Red Brick 31-2, 31-6
 - for Sybase ASE 37-3
 - for Sybase IQ 38-3
 - for Teradata 39-6
 - for UniVerse 40-2
- connection script (tnfocus.scr) files for Bull 4-3
- CONNECTION= attribute for Enterprise Java Beans (EJB) 8-5
- contiguous keys for RMS 32-13
- copying files
 - J.D. Edwards 16-4
- COVERAGE record type for IDMS/DB 10-9
- CREATE PROCEDURE command for Informix 13-21
- CREATE SYNONYM command 7-11, 8-9, 13-8, 40-10
 - for Adabas 2-29
 - for Enterprise Java Beans (EJB) 8-9
 - for Essbase 9-5
 - for IMS 12-16
 - for Informix 13-8
 - for Microsoft Access 18-5
 - for Microsoft SQL Server 19-8
 - for Microsoft SQL Server Analytical Engine 20-7
 - for MQSeries 22-4
 - for MUMPS 23-4
 - for ODBC 26-5
 - for Oracle 27-10
 - for Rdb 30-5
 - for Red Brick 31-6
 - for Sybase ASE 37-7
 - for Sybase IQ 38-7
 - for UniVerse 40-2
- Global Format Buffer ID (GFBID) parameter for Adabas 2-32

Index

- creating synonyms 2-22, 13-8
 - for Adabas 2-22 to 2-23
 - for Bull 4-10
 - for DB2 7-11
 - for Enterprise Java Beans (EJB) 8-6
 - for Essbase 9-4 to 9-5
 - for IDMS/SQL 11-4
 - for Informix 13-8
 - for Informix using the Web Console 13-8
 - for Informix using Web Console 13-8
 - for Microsoft Access 18-2
 - for Microsoft SQL Server 19-8
 - for Microsoft SQL Server Analytical Engine 20-6
 - for Microsoft SQL Server using Web Console 19-8 to 19-9
 - for MQSeries 22-3 to 22-4
 - for MUMPS 23-3 to 23-4
 - for Nucleus 25-6 to 25-7, 25-9
 - for ODBC 26-5
 - for Oracle 27-10
 - for Oracle using Web Console 27-10
 - for Progress 29-7
 - for Rdb 30-5
 - for Rdb using Web Console 30-5 to 30-6
 - for Red Brick 31-6
 - for Red Brick using Web Console 31-6 to 31-7
 - for SAP BW 33-14
 - for SAP/R3 34-18
 - for Siebel 35-10
 - for Sybase ASE 37-7
 - for Sybase ASE using the Web Console 37-7
 - for Sybase IQ 38-7
 - for Teradata 39-6
 - for Teradata using Web Console 39-7
 - for UniVerse 40-2
 - for XML 41-4, 41-6 to 41-7
- creating synonyms manually 2-22, 19-8
 - for Adabas 2-22
 - for Bull 4-12
 - for DB2 7-15
 - for Enterprise Java Beans (EJB) 8-8
 - for Essbase 9-4
- creating synonyms manually (*continued*)
 - for IDMS/SQL 11-4
 - for Microsoft SQL Server 19-8, 19-12
 - for Microsoft SQL Server Analytical Engine 20-6
 - for MQSeries 22-3
 - for MUMPS 23-3
 - for Nucleus 25-10
 - for ODBC 26-5
 - for Oracle 27-10
 - for Rdb 30-8 to 30-9
 - for SAP/R3 34-28
 - for Siebel 35-14
 - for Sybase ASE 37-11
 - for Teradata 39-10
 - for XML 41-4
- creating virtual fields 1-13 to 1-14
- CRFILE keyword for IDMS/DB 10-19
- cross-century dates 1-14 to 1-17
- cross-referenced files in Adabas 2-70
- cross-referencing data sources for DATACOM 6-17
- cube-slicing logic for SAP BW 33-24
- cursor types for Microsoft SQL Server 19-23
- customization 3-13
 - for Adabas Data Adapter 2-72
 - for Allbase Data Adapter 3-13
 - for DB2 Data Adapter 7-24
 - for Essbase Data Adapter 9-12
 - for IDMS/DB Data Adapter 10-61
 - for IDMS/SQL Data Adapter 11-15
 - for Informix Data Adapter 13-2
 - for Microsoft Access Data Adapter 18-17
 - for Microsoft SQL Server Analytical Engine Data Adapter 20-8
 - for Microsoft SQL Server Data Adapter 19-23
 - for Nucleus Data Adapter 25-14
 - for ODBC Data Adapter 26-13
 - for Oracle Data Adapter 27-25
 - for Progress Data Adapter 29-17
 - for Red Brick Data Adapter 31-15

customization (*continued*)

- for Sybase ASE 37-19
- for Sybase IQ 38-19
- for Teradata Data Adapter 39-17
- for UniVerse Data Adapter 40-13

CV (Central Version) mode for IDMS/DB 10-33

D

data adapters 1-1 to 1-4

Data Definition Language commands for UniVerse 40-13

Data Definition Module (DDM) for Adabas 2-14

- data formats 2-41
- for Adabas 2-41
 - for IDMS/SQL 11-12
 - for MQSeries 22-5
 - for MUMPS 23-6
 - for Siebel 35-17
 - for XML 41-9

data location for PeopleSoft 28-3

Data Management Language (DML) for IDMS/DB 10-3

Data Manipulation Language (DML) 1-3

- data sources 1-6, 1-8
- describing 1-6

data sources for DATACOM 6-2

- data types 3-4
- for Allbase 3-4
 - for DB2 7-11
 - for Enterprise Java Beans (EJB) 8-11
 - for Informix 13-8
 - for Microsoft Access 18-5
 - for Microsoft SQL Server 19-8
 - for My SQL 24-6
 - for Nucleus 25-6
 - for ODBC 26-5
 - for Oracle 27-10

data types (*continued*)

- for Progress 29-7
- for Rdb 30-5
- for Red Brick 31-6
- for SAP/R3 34-31
- for Sybase ASE 37-7
- for Sybase IQ 38-7
- for UniVerse 40-2, 40-11

database key for IDMS/DB 10-24, 10-84

database numbers for Adabas 2-76

Database Resource Adapter (DRA) for IMS 12-2 to 12-3

- Database Servers 27-5
- for Oracle 27-5
 - for Progress 29-3
 - for Siebel 35-2

database tables for ODBC 26-5

DATACOM Data Adapter 6-1 to 6-2

- Access Files 6-12, 6-16, 6-18
- application requests 6-1
- Boolean Selection capabilities 6-2
- Compound Boolean Selection 6-22
- configuring using the Web Console 6-2
- connection attributes 6-2
- cross-referencing between data sources 6-17
- Data Dictionary Database 6-18
- data sources 6-2
- DBID attribute 6-13
- element logical record security 6-14
- element-level security 6-2
- field attributes for Master Files 6-10
- header logical record 6-13
- IXFLD attribute 6-13
- KEYFIELD attribute 6-13
- KEYNAME attribute 6-13
- mapping considerations 6-2
- Master File definitions 6-5
- Master Files 6-9, 6-15, 6-18
- metadata 6-8
- multi-file structures 6-14, 6-18

Index

- DATAACOM Data Adapter (*continued*)
 - preparing the environment 6-2
 - record retrieval commands 6-22
 - segment attributes for Master Files 6-10
 - segment logical record 6-13
 - SEGNAME attribute 6-13
 - Sequential Retrieval Commands 6-22
 - server terminology 6-5
 - TABlename attribute 6-13
 - TRACE attribute 6-13
 - User Requirements Table (URT) 6-2
 - USERTABLE attribute 6-13
- DATAACOM Data Dictionary 6-4 to 6-5
- DATAACOM data structures 6-3, 6-5
 - and Access Files 6-2
 - and Master Files 6-2
 - elements 6-3
 - Employee Address Element (ADEMP) 6-4
 - Employee Record Element (EMDTA) 6-4, 6-7
 - fields 6-3
 - records 6-3
- date fields for XML 41-10
- DB2 1-18
- DB2 Data Adapter 7-1
 - Access File keywords 7-11
 - configuration 7-3
 - connection attributes 7-3
 - CREATE SYNONYM command 7-11
 - creating synonyms 7-11
 - creating synonyms manually 7-15
 - creating synonyms using the Web Console 7-11
 - customizing 7-24
 - data types 7-11
 - environment preparation 7-2
 - large character data types 7-11
 - metadata 7-11
 - preparing the environment 7-2
 - synonyms 7-11
 - variable-length data types 7-11
 - XA support 7-3
- DB2 data types 7-11
- DB2 interface A-4
- DB2 Universal Database CLI Adapter for PeopleSoft 28-8
- DBCTL environment for IMS 12-6
- DBMS for MODEL 204 21-30, 21-35
- DBNAME parameter for IDMS/DB 10-91
- DBNO attribute for Adabas 2-50
- DBSPACE command for IDMS/SQL 11-17
- ddname CSIPARM for Supra 36-3
- ddname CSISYSIN for Supra 36-3
- default connections 25-3
 - for Microsoft SQL Server Analytical Engine 20-4 to 20-5
 - for Nucleus 25-3
 - for Rdb 30-2, 30-4 to 30-5
 - for Red Brick 31-2
 - for Siebel 35-9
 - for UniVerse 40-2
- default passwords for Adabas 2-11
- default settings for MODEL 204 21-36
- DEFCENT parameter 1-15
- DEFINE attribute for RMS 32-47
- DEFINE command 1-13 to 1-14
- defining PSB resources for IMS 12-8
- DEPARTMENT record type for IDMS/DB 10-9
- descendant records for Adabas 2-105
- descendant segments for IDMS/DB 10-66, 10-87
- describing files manually for RMS 32-2
- describing subschemas in IDMS/DB 10-37
- DESCRIPTION attribute 1-22 to 1-23

- DESCRIPTION attribute for RMS 32-48
 - descriptors for Adabas 2-19, 2-54
 - Descriptors (DE) 2-19
 - Subdescriptors (NOP) 2-19
 - Superdescriptors (SPR) 2-19
 - DICTNAME parameter for IDMS/DB 10-91
 - dimension levels for SAP BW 33-21
 - Direct Passthru (DPT) 1-3
 - discontiguous keys for RMS 32-14
 - DML (Data Management Language) for IDMS/DB 10-3
 - DML (Data Manipulation Language) 1-3
 - DOC Synonyms for SAP/R3 34-18
 - documenting columns 1-22 to 1-23
 - documenting tables 1-22
 - DPT (Direct Passthru) 1-3
 - DRA (Database Resource Adapter) for IMS 12-2 to 12-3
 - DROP SYNONYM command for Bull 4-13
 - dynamic database numbers for Adabas 2-76
 - dynamic dimensions for SAP BW 33-27 to 33-28
 - dynamic joins for MODEL 204 21-7
- E**
-
- EDAPATH for J.D. Edwards 16-5
 - EDASTART for Supra 36-3
 - Edit Synonyms window for Enterprise Java Beans (EJB) 8-8
 - editing ISTART JCL for IDMS/DB 10-2
 - element-level security for DATACOM 6-2
 - embedded data for RMS 32-30
 - embedded JOINS for Adabas 2-59, 2-61
 - Embedded Joins for MODEL 204 21-9
 - embedded repeating data for RMS 32-23
 - EMPLOYEE record type for IDMS/DB 10-7
 - EMPOSITION record type for IDMS/DB 10-6 to 10-7
 - EMPSCHM database schema for IDMS/DB 10-40
 - Enterprise Java Beans (EJB) Data Adapter 8-1 to 8-2
 - Access File keywords 8-10
 - Access Files 8-6
 - application requests 8-1
 - BEA WebLogic Application Server 8-2, 8-4
 - configuring 8-2, 8-5
 - connection scope 8-5 to 8-6
 - CONNECTION= attribute 8-5
 - CREATE SYNONYM command 8-9
 - creating synonyms 8-6
 - creating synonyms manually 8-8
 - creating synonyms using the Web Console 8-6
 - data types 8-11
 - Edit Synonyms window 8-8
 - IBM WebSphere Application Server 8-2
 - Master Files 8-6
 - metadata 8-6
 - modifying synonyms 8-8
 - modifying synonyms using the Web Console 8-8
 - preparing the Web Application Server environment 8-2
 - SET CONNECTION_ATTRIBUTES command 8-2
 - entry segment retrieval 2-95, 10-83
 - for Adabas 2-95
 - for IDMS/DB 10-83
 - environment preparation 14-2, 37-2
 - for Allbase 3-2
 - for DB2 7-2
 - for Informix 13-20
 - for Ingres II 14-2
 - for Interplex 15-2
 - for Lawson 17-2

environment preparation (*continued*)

- for Microsoft Access 18-2
- for Microsoft SQL Server 19-2
- for Nucleus 25-2
- for ODBC 26-2
- for Oracle 27-2
- for PeopleSoft 28-2, 28-6
- for Progress 29-2
- for Rdb 30-2
- for Red Brick 31-2
- for SAP BW 33-2
- for Siebel 35-2
- for Sybase ASE 37-2
- for Sybase IQ 38-2
- for Teradata 39-2
- for UniVerse 40-2

environment variables 35-3

- for MQSeries 22-2
- for Siebel 35-3

Essbase Data Adapter 9-1 to 9-2

- Access Files 9-6
- ALIAS attribute 9-15 to 9-16
- application requests 9-1
- configuring 9-2
- connection attributes 9-3
- CREATE SYNONYM command 9-5
- creating synonyms 9-4 to 9-5
- creating synonyms manually 9-4
- customizing 9-12
- field names 9-6
- JOIN command 9-12
- Master File 9-9 to 9-10
- MAXROWS setting 9-17
- measures dimension 9-10
- metadata 9-4
- missing data 9-22 to 9-23
- Non-Aggregated Fields 9-18
- non-aggregated members 9-19
- parent/child view 9-7 to 9-8, 9-12
- preparing the environment 9-2
- preparing the environment on UNIX 9-2

Essbase Data Adapter (*continued*)

- preparing the environment on Windows NT/2000 9-2
- PRINT command 9-12
- scenario dimensions 9-9
- security 9-2
- SET CONVERSION command 9-12
- SET MEASURE command 9-10
- SET SCENARIO command 9-9
- Shared Members 9-20
- substitution variables 9-24
- SUM command 9-12
- SUPEMPTY setting 9-23
- SUPMISSING setting 9-22
- SUPSHARE setting 9-20
- SUPZERO setting 9-22
- Time Series reporting 9-17
- UDAs (User-Defined Attributes) 9-8
- user authentication 9-2
- User-Defined Attributes (UDAs) 9-8
- zero values 9-22 to 9-23

executing XML documents with XML XFOCUS 41-11

execution results for PeopleSoft 28-47

EXPERTISE record type for IDMS/DB 10-7

Extended Catalog

- accessing metadata 1-17

F

FBTL buffer for MODEL 204 21-34

FDEFCENT setting 1-16 to 1-17

FETCH attribute for Adabas 2-53, 2-75

FETCHSIZE attribute for Adabas 2-53, 2-75

FIELD attribute for RMS 32-49

field attributes 1-9, 11-7

- for IDMS/DB 10-19 to 10-20
- for IDMS/SQL 11-7
- for MODEL 204 21-17, 21-30
- for RMS 32-4, 32-41 to 32-42

- field declarations 1-8 to 1-9
 - field declarations for MODEL 204 21-29
 - field definition tables (FDT) for Adabas 2-15
 - field names 9-6
 - for Essbase 9-6
 - for RMS 32-5
 - for SAP BW 33-13
 - SAP BW 33-13
 - FIELDNAME attribute 1-9, 2-41, 21-18
 - for Adabas 2-41
 - for IDMS/DB 10-19
 - for MODEL 204 21-18
 - for RMS 32-5, 32-49
 - FIELDTYPE attribute for RMS 32-49
 - FILE attribute 32-50
 - for MODEL 204 21-25
 - for RMS 32-50
 - file attributes 21-14
 - for MODEL 204 21-14
 - for RMS 32-2, 32-39
 - file declarations 1-8
 - file inversion for IDMS/DB 10-73 to 10-74
 - file locking for RMS 32-34
 - file retrieval for IDMS/DB 10-78
 - FILEDEF command 22-2
 - for MQSeries 22-2
 - for XML 41-2 to 41-3
 - FILEDEF command for MQSeries 22-2
 - FILENAME attribute 1-8
 - FILENAME attribute for RMS 32-2, 32-50
 - FILENO attribute for Adabas 2-50
 - FILESUFFIX attribute for RMS 32-52
 - FIND call for Adabas 2-84
 - FIND navigation for Adabas 2-102
 - FIX files for RMS 32-17
 - fixed-length records in Adabas files 2-20
 - flat files 5-4
 - specifying location on an HFS file system 5-4
 - specifying location on OS/390 5-4
 - specifying locations on UNIX and Windows NT 5-3
 - FM Synonyms for SAP/R3 34-18, 34-23
 - FOCPATH command for Bull 4-15 to 4-16
 - foreign messages for MQSeries 22-4
 - formulas for SAP BW 33-12
 - frequently asked questions
 - J.D. Edwards 16-27
 - functional limitations for Bull 4-3
 - FYRTHRESH setting 1-16 to 1-17
- G**
-
- GATEKEPR Service Block for Supra 36-5
 - GCOS7 environment 4-8, 4-16
 - file system syntax 4-8
 - for Bull 4-2
 - GCOS8 environment 4-3, 4-8
 - accessing 4-3
 - file system syntax 4-8
 - for Bull 4-2
 - generating DBA
 - for Lawson 17-6
 - generic parameters for Adabas 2-9
 - GFBID (Global Format Buffer ID) for Adabas 2-32
 - Global Format Buffer ID (GFBID) for Adabas 2-32
 - global settings for PeopleSoft 28-14

Index

GROUP attribute 2-45
for Adabas 2-45, 2-55, 2-70
for RMS 32-12 to 32-14, 32-50

group fields for Adabas 2-87

GROUP keyword for IDMS/DB 10-24

H

HEADER logical record for Supra 36-14

HELPMESSAGE attribute 1-25 to 1-26

hierarchies 33-27
for SAP BW 33-27
for SAP/R3 33-33 to 33-34
SAP BW 33-27

hierarchy variables for SAP BW 33-21

I

IBM WebSphere Application Server for Enterprise
Java Beans (EJB) 8-2
connection attributes 8-4

IDMS/DB Data Adapter 10-1, 10-3
Access Files 10-31, 10-54, 10-59
ACTUAL attribute 10-22
ALIAS attribute 10-21
application requests 10-1
area sweep record retrieval 10-87
ASF (Automatic System Facility) 10-14
Automatic System Facility (ASF) 10-14
bill-of-materials structures 10-10
CALC field record retrieval 10-84
CALC fields 10-13
CALC-based retrieval 10-89
Central Version (CV) mode 10-33
CHECK FILE command 10-63
common member structure 10-8
common owner structure 10-7
configuring 10-2
connection attributes 10-2
COVERAGE record type 10-9
CRFILE keyword 10-19

IDMS/DB Data Adapter (*continued*)
customizing 10-61
CV (Central Version) mode 10-33
Data Management Language (DML) 10-3
database key 10-24
database key record retrieval 10-84
DBNAME parameter 10-91
DEPARTMENT record type 10-9
descendant segments 10-66, 10-87
describing subschemas 10-37
DICTNAME parameter 10-91
DML (Data Management Language) 10-3
editing ISTART JCL for 10-2
EMPLOYEE record type 10-7
EMPOSITION record type 10-6 to 10-7
EMPSCHM database schema 10-40
entry segment retrieval 10-83
EXPERTISE record type 10-7
field attributes 10-19 to 10-20
FIELDNAME attribute 10-19
file attributes 10-17
file inversion 10-73 to 10-74
file retrieval 10-78
GROUP keyword 10-24
IF tests 10-66 to 10-67
implementing parent/descendant relationships
with SELECT 10-15
index declaration keywords 10-39
index declarations 10-31
index fields 10-13
index record retrieval 10-85
index-based retrieval 10-89
intra-record structures 10-26
INVOICE record type 10-12
JOB record type 10-6
JOIN command 10-77 to 10-78
joining data sources 10-76
joining Master Files 10-75
KL segments 10-25
KLU segments 10-25
limiting record retrieval 10-67
Logical Record Facility (LRF) 10-3, 10-14, 10-90

IDMS/DB Data Adapter (*continued*)

- Logical Record Facility (LRF) record retrieval 10-90
- Logical Record Facility (LRF) records 10-37
- Logical Record Facility (LRF) records as descendants 10-15
- Logical Record Facility (LRF) subschema 10-59
- loop structures 10-12
- LRF (Logical Record Facility) 10-3, 10-14, 10-90
- LRF (Logical Record Facility) record retrieval 10-90
- LRF (Logical Record Facility) records 10-37
- LRF (Logical Record Facility) records as descendants 10-15
- LRF (Logical Record Facility) subschema 10-59
- mapping concepts in 10-3
- Master Files 10-17
- metadata 10-17
- multi-member sets 10-9
- network relationships 10-16
- network subschema 10-48 to 10-49, 10-54
- non-unique descendant segments 10-69, 10-83
- OCCURS attribute 10-19
- OCCURS segment 10-26
- ORDER field 10-29 to 10-30
- overriding parameters 10-91
- PARENT keyword 10-18
- POSITION attribute 10-19
- POSITION keyword 10-28
- preparing the environment 10-2
- record retrieval 10-67, 10-87
- record retrieval intervals 10-78, 10-83
- repeating groups 10-26
- reporting from joined structures 10-78
- retrieval subtree 10-62 to 10-63, 10-65, 10-79
- sample file descriptions 10-40
- sample LRF (Logical Record Facility) record 10-59
- screening conditions 10-81 to 10-82
- screening criteria for SET ALL = ON command 10-71
- segment attributes 10-18

IDMS/DB Data Adapter (*continued*)

- segment declaration keywords 10-34, 10-37
- segment declarations 10-31, 10-34
- SEGNAME attribute 10-18
- selection criteria 10-65
- selective ALL prefix 10-72
- SEQFIELD parameter 10-86
- Sequential Processing Facility (SPF) 10-39
- Sequential Processing Facility (SPF) indexes 10-60
- SET ALL = OFF command 10-70
- SET ALL = ON command 10-71
- set-based relationships 10-5
- set-based retrieval 10-88
- short paths 10-68 to 10-69, 10-82 to 10-83
- simple sets 10-6, 10-8
- sort paths 10-74
- SPF (Sequential Processing Facility) 10-39
- SPF (Sequential Processing Facility) indexes 10-60
- STRUCTURE record type 10-10
- subschemas 10-36
- tracing 10-92
- unique descendant segments 10-69, 10-82
- unique segments 10-68, 10-82
- unique segments in retrieval subtree 10-65
- unique segments in retrieval subtrees 10-64
- USAGE attribute 10-22

IDMS/SQL Data Adapter 11-1 to 11-2

- Access Files 11-4, 11-9
- application requests 11-1
- configuring 11-2
- CONNECT command 11-2
- connection scope 11-3
- creating synonyms 11-4
- creating synonyms manually 11-4
- customizing 11-15
- data formats 11-12
- DBSPACE command 11-17
- IXSPACE command 11-17
- Master Files 11-4 to 11-5, 11-7

Index

IDMS/SQL Data Adapter (*continued*)

- metadata 11-3
- MISSING attribute 11-9
- numeric columns 11-13
- PASSRECS 11-18
- precision attribute 11-13
- precision value 11-14
- preparing the environment 11-2
- primary key 11-11
- scale values 11-13 to 11-14
- segment attributes 11-6
- segment declarations 11-10
- session commands 11-3
- SET TRANSACTION command 11-15
- setting user-specific schema names 11-15

IDOC Synonyms for SAP/R3 34-21

IF tests for IDMS/DB 10-66 to 10-67

IFAM2 applications for MODEL 204 21-28

IFAMCHNL attribute for MODEL 204 21-23

IFS and QSYS file systems for J.D. Edwards 16-4

implementing parent/descendant relationships with SELECT for IDMS/DB 10-15

IMS Data Adapter 12-1

- Access Files 12-10
- application requests 12-1
- configuring 12-9
- CREATE SYNONYM command 12-16
- creating synonyms 12-10
- creating synonyms manually 12-15
- creating synonyms using the Web Console 12-10
- Database Resource Adapter (DRA) 12-2
- DBCTL environment 12-1, 12-6
- DRA (Database Resource Adapter) 12-2
- dynamic settings for PSBs 12-9
- keywords 12-4
- Master Files 12-10
- metadata 12-10
- migrating from an existing MVS Server 12-17

IMS Data Adapter (*continued*)

- preparing the environment 12-2
- PSB resources 12-8
- security 12-7
- startup tables 12-2 to 12-3
- Web Console 12-9

IMS DBCTL environment 12-1

INDEX attribute 2-45

- for Adabas 2-45
- for RMS 32-49

index declaration keywords for IDMS/DB 10-39

index fields for IDMS 10-13

index record retrieval DMS/DB 10-85

INDEX=I attribute for Adabas 2-54

index-based retrieval for IDMS/DB 10-89

indexed files for RMS 32-12, 32-37

InfoCubes for SAP BW 33-8 to 33-9, 33-18

Informix Client SDK software 13-2

Informix Data Adapter 13-1

- Access File keywords 13-8
- ANSI-compliant data sources 13-3
- application requests 13-1
- authentication 13-4
- AUTODISCONNECT command 13-4
- Client SDK software 13-2
- configuring 13-4
- connection attributes 13-4
- connection scope 13-4
- CREATE PROCEDURE command 13-21
- CREATE SYNONYM command 13-8
- creating synonyms 13-8
- creating synonyms using the Web Console 13-8
- customizing 13-20
- data types 13-8
- Informix SE 13-3
- large character data types 13-8
- mapping data types 13-8

- Informix Data Adapter (*continued*)
 - metadata 13-8
 - numeric columns 13-8
 - precision values 13-17
 - preparing the environment 13-20
 - privileges 13-3
 - remote servers 13-3
 - rows 13-20
 - scale values 13-17
 - SET commands 13-4
 - SET CONNECTION_ATTRIBUTES command 13-4
 - SET CONVERSION command 13-8
 - SET DEFAULT_CONNECTION command 13-4
 - SET parameters 13-20
 - SQL Passthru 13-21
 - stored procedures 13-21
 - uncommitted transactions 13-4
 - UNIX privileges 13-3
 - XA support 13-4
 - Informix SE 13-3
 - Informix SQL ID 13-2 to 13-3
 - Ingres II Data Adapter 14-1
 - application requests 14-1
 - preparing the environment 14-2
 - INSERTSIZE command 19-23
 - for Microsoft SQL Server 19-23
 - for Oracle 27-25
 - installation directory for C-ISAM 5-2
 - installation prerequisites
 - J.D. Edwards 16-2
 - Interfaces file for Sybase ASE 37-2
 - interfile relationships for MODEL 204 21-6
 - Internal Sequence Number (ISN) for Adabas 2-30 to 2-31
 - Interplex Data Adapter 15-1
 - application requests 15-1
 - configuring 15-2
 - precision values 15-10
 - preparing the environment 15-2, 15-13
 - scale values 15-10
 - iWay Data Adapter Administration for UNIX, Windows, OpenVMS, OS/400, OS/390 and z/OS
 - intra-record structures for IDMS/DB 10-26
 - inverted lists for Adabas 2-14
 - INVOICE record type for IDMS/DB 10-12
 - iWay for Bull RTE (Run Time Environment) 4-3, 4-17
 - connection script (tnfocus.scr) files 4-3
 - REMOTE EXEC facility 4-17
 - REMOTE PUT/GET facility 4-17
 - IXFLD attribute 2-59
 - for Adabas 2-59
 - for MODEL 204 21-26
 - IXSPACE command for IDMS/SQL 11-17
- J**
-
- J.D. Edwards Data Adapter 16-1
 - Access Files 16-21
 - APP 16-6
 - APP MAP 16-6
 - application requests 16-1
 - copying files 16-4
 - EDAPATH 16-5
 - frequently asked questions 16-27
 - IFS and QSYS file systems 16-4
 - installation prerequisites 16-2
 - linking files 16-5
 - major system codes 16-24
 - Master Files 16-21
 - metadata 16-6
 - QSYS and IFS file systems 16-4
 - report library 16-23
 - security 16-22
 - setup 16-2
 - software requirements 16-2
 - tracing 16-23
 - unloading the CD-ROM 16-3
 - J.D. Edwards major system codes 16-24
 - Java 2 SDK for Siebel 35-2
 - on UNIX 35-4
 - on Windows 35-3

JOB record type for IDMS/DB 10-6

JOIN command 2-85

for Adabas 2-85

for Essbase 9-12

for IDMS/DB 10-77 to 10-78

for MODEL 204 21-7

joined files for Adabas 2-85

joining data sources in IDMS/DB 10-76

joining Master Files in IDMS/DB 10-75

joins support for SAP/R3 34-34

JSCOM3 listener for Siebel 35-4

K

KEY attribute for RMS 32-50

key fields for MODEL 204 21-30

key figures for SAP BW 33-11

keyed (indexed) files for RMS 32-12, 32-33

KEYFLD attribute for Adabas 2-59

KEYFLD attribute for MODEL 204 21-26

KEYORDER attribute 1-12

KEYS attribute 1-12

keys for RMS 32-12

keywords 25-6, 40-10

for IMS 12-4

for Nucleus 25-6

for Rdb 30-5

for Sybase IQ 38-7

KL segments for IDMS/DB 10-25

KLU segments for IDMS/DB 10-25

L

large character data types 13-8

for DB2 7-11

for Informix 13-8

for Microsoft Access 18-5

for Microsoft SQL Server 19-8

for Oracle 27-10

for Progress 29-7

for Sybase ASE 37-7

Lawson Data Adapter 17-1

configuration 17-4

connection attributes 17-4 to 17-5

generating DBA 17-6

metadata 17-5

preparing the environment 17-2

security 17-2, 17-6

WHERE clauses 17-6

LIBRARY file for RMS 32-20 to 32-21

linking files

J.D. Edwards 16-5

listeners for Siebel 35-4

locked records 21-35

for MODEL 204 21-35

for RMS 32-35

locking files for RMS 32-34

LOCKMODE setting for RMS 32-35

Logical Record Facility (LRF) for IDMS/DB 10-3, 10-14

record retrieval 10-90

records as descendants 10-15

sample records 10-59

subschemas 10-55

Logical Record Facility (LRF) record retrieval for IDMS/DB 10-90

login wait time for Microsoft SQL Server 19-23

- LOGINTIMEOUT command for Microsoft SQL Server 19-23
- loop structures for IDMS/DB 10-12
- LRF (Logical Record Facility) for IDMS/DB 10-3, 10-14
 - record retrieval 10-90
 - records as descendants 10-15
 - sample records 10-59
 - subschemas 10-55
- LRF (Logical Record Facility) record retrieval for IDMS/DB 10-90
- M**

- M204IN SET MAXMBUFF command for MODEL 204 21-33
- M204IN SET MISSING command for MODEL 204 21-34
- M204IN SET SINGLETHREAD command for MODEL 204 21-36
- MAINTAIN operations for RMS 32-57
- manually describing files for RMS 32-2
- MAPFIELD value for RMS 32-31
- mapping considerations 6-2
 - for DATACOM 6-2
 - for MODEL 204 21-3
 - for Supra 36-9, 36-11
- mapping data types 13-8
 - for Informix 13-8
 - for Microsoft Access 18-5
 - for Microsoft SQL Server 19-8
 - for Nucleus 25-6
 - for ODBC 26-5
 - for Oracle 27-10
 - for Progress 29-7
 - for Red Brick 31-6
 - for Sybase ASE 37-7
 - for Sybase IQ 38-7
- mapping descriptors in Adabas 2-54
- mapping rules for MODEL 204 21-13
- MAPVALUE field for RMS 32-31
- Master File attributes 1-21 to 1-23, 1-25
- Master Files 1-6, 1-8 to 1-9, 1-11, 1-16, 2-29
 - attributes for Adabas 2-26
 - attributes for DATACOM 6-9
 - attributes for IDMS/DB 10-17
 - attributes for IDMS/SQL 11-5
 - attributes for RMS 32-39
 - declarations for DATACOM 6-9
 - field attributes for DATACOM 6-10
 - field attributes for IDMS/SQL 11-7
 - file attributes for Adabas 2-36 to 2-37
 - for Adabas 2-22, 2-29, 2-35
 - for Bull 4-10
 - for DATACOM 6-2, 6-5
 - for Enterprise Java Beans (EJB) 8-6
 - for Essbase 9-9 to 9-10
 - for IDMS/DB 10-17
 - for IDMS/SQL 11-4 to 11-5
 - for IMS 12-10
 - for MODEL 204 21-14
 - for MQSeries 22-3
 - for MUMPS 23-3
 - for RMS 32-39
 - for SAP BW 33-20
 - for SAP/R3 34-18
 - for Supra 36-11
 - for XML 41-2, 41-4
 - identifying for IDMS/DB 10-17
 - J.D. Edwards 16-21
 - joining for IDMS/DB 10-75
 - mapping descriptors in Adabas 2-54
 - multiple DATACOM logical files in 6-15
 - root segment in Adabas 2-38
 - segment attributes for Adabas 2-37
 - segment attributes for DATACOM 6-10
 - segment attributes for IDMS/SQL 11-6
 - Siebel 35-10

Index

- MAXROWS setting for Essbase 9-17
- measures dimension for Essbase 9-10
- member variables for SAP BW 33-21
- metadata 2-21, 3-4, 4-9, 6-8, 8-6, 10-17, 41-4
 - for Adabas 2-21
 - for Allbase 3-4
 - for Bull 4-9
 - for DATACOM 6-8
 - for DB2 7-11
 - for Enterprise Java Beans (EJB) 8-6
 - for Essbase 9-4
 - for IDMS/DB 10-17
 - for IDMS/SQL 11-3
 - for IMS 12-10
 - for Informix 13-8
 - for Lawson 17-5
 - for Microsoft Access 18-5
 - for Microsoft SQL Server 19-8
 - for Microsoft SQL Server Analytical Engine 20-6
 - for MODEL 204 21-14
 - for MQSeries 22-2
 - for MUMPS 23-3
 - for Nucleus 25-6
 - for ODBC 26-5
 - for Oracle 27-10
 - for PeopleSoft 28-25 to 28-26, 28-35
 - for Progress 29-7
 - for Rdb 30-5
 - for Red Brick 31-6
 - for SAP BW 33-13, 33-17, 33-19
 - for SAP/R3 34-18
 - for Siebel 35-9
 - for Supra 36-11
 - for Sybase ASE 37-7
 - for Sybase IQ 38-7
 - for UniVerse 40-2
 - for XML 41-4
 - J.D. Edwards 16-6
 - SAP BW 33-17
- Metadata calls 1-18
 - Metadata Services
 - Calls 1-17
 - User-defined Metadata 1-18 to 1-19
 - Micro Focus files for C-ISAM 5-3
 - Microsoft Access Data Adapter 18-1
 - application requests 18-1
 - configuring 18-2
 - connection attributes 18-2
 - connection scope 18-2
 - CREATE SYNONYM command 18-5
 - creating synonyms 18-2
 - creating synonyms using the Web Console 18-2
 - customizing 18-17
 - data types 18-5
 - large character data types 18-5
 - metadata 18-5
 - ODBC Driver Manager 18-2
 - precision values 18-14
 - preparing the environment 18-2
 - scale values 18-14
 - SET commands 18-2, 18-5
 - SET CONNECTION_ATTRIBUTES command 18-2
 - uncommitted transactions (LUW) 18-2
 - Microsoft Access data types 18-5
 - Microsoft SQL Server Analytical Engine Data Adapter 20-1
 - application requests 20-1
 - configuring 20-3
 - connection attributes 20-3
 - CREATE SYNONYM command 20-7
 - creating synonyms 20-6
 - creating synonyms manually 20-6
 - customizing 20-8
 - default connections 20-4 to 20-5
 - environment variables 20-2
 - metadata 20-6
 - NONBLOCK mode 20-8
 - preparing the environment 20-2
 - preparing the environment on Windows NT/2000 20-2

- Microsoft SQL Server Analytical Engine Data Adapter (*continued*)
 - security 20-2
 - SET CONNECTION_ATTRIBUTES command 20-4
 - SLICEROPTIMIZATION command 20-9
 - user authentication 20-2
- Microsoft SQL Server Data Adapter 19-1
 - Access File keywords 19-8
 - accessing remotely 19-2
 - application requests 19-1
 - array retrieval 19-23
 - AUTODISCONNECT command 19-3
 - block size 19-23
 - configuring 19-3
 - connection attributes 19-3, 19-5
 - connection scope 19-3
 - CREATE SYNONYM command 19-8
 - creating synonyms 19-8
 - creating synonyms manually 19-12
 - creating synonyms using the Web Console 19-9
 - cursor types 19-23
 - customizing 19-23
 - data types 19-8
 - explicit authentication 19-3
 - INSERTSIZE command 19-23
 - large character data types 19-8
 - login wait time 19-23
 - LOGINTIMEOUT command 19-23
 - metadata 19-8
 - National Language Support (NLS) 19-8
 - NLS (National Language Support) 19-8
 - numeric columns 19-8, 19-20
 - password passthru authentication 19-3
 - precision values 19-20 to 19-21
 - preparing the environment 19-2
 - read-only fields 19-8
 - scale values 19-20 to 19-21
 - security 19-3, 19-5
 - SET CONNECTION_ATTRIBUTES command 19-3
 - SET TRANSACTIONS command 19-23
 - SQL Passthru 19-23
 - stored procedures 19-23
- Microsoft SQL Server Data Adapter (*continued*)
 - synonyms 19-8, 19-11
 - trusted authentication 19-3
 - uncommitted transactions (LUW) 19-3
 - XA support 19-2
 - XA Transaction Management feature 19-2
- migrating from MVS Servers on IMS 12-17
- MISSING attribute 1-9, 1-11
- MISSING attribute for IDMS/SQL 11-9
- missing data
 - suppressing in Essbase 9-22 to 9-23
- missing data for MODEL 204 21-34 to 21-35
- missing non-unique segments for Adabas 2-79
- MODEL 204 Data Adapter 21-1 to 21-3
 - ACCESS attribute 21-28
 - Access Files 21-21, 21-31
 - ACCOUNT attribute 21-23
 - account declaration 21-23
 - ACCOUNTPASS attribute 21-23
 - ACTUAL attribute 21-19
 - ALIAS attribute 21-4, 21-18
 - application requests 21-1
 - authentication 21-2
 - buffer capacity 21-33
 - configuring 21-2
 - DBMS 21-30, 21-35
 - default settings 21-36
 - dynamic joins 21-7
 - Embedded Joins 21-9
 - environment variables 21-2
 - FBTL buffer 21-34
 - field attributes 21-17, 21-30
 - field declarations 21-29
 - FIELDNAME attribute 21-18
 - FILE attribute 21-25
 - file attributes 21-14
 - IFAM2 applications 21-28
 - IFAMCHNL attribute 21-23
 - interfile relationships 21-6

Model 204 Data Adapter (*continued*)

- IXFLD attribute 21-26
- JOIN command 21-7
- key fields 21-30
- KEYFLD attribute 21-26
- locked records 21-35
- M204IN SET MAXMBUFF command 21-33
- M204IN SET MISSING command 21-34
- M204IN SET SINGLETHREAD command 21-36
- mapping considerations 21-3
- mapping rules 21-13
- Master File 21-14
- metadata 21-14
- missing data 21-34 to 21-35
- MODEL 204 DBMS 21-35
- multiply occurring fields 21-5
- OCCURS attributes 21-19
- OCCURS segment 21-28
- ORDER field 21-20
- PARENT attribute 21-16
- PASS attribute 21-25
- preparing the environment 21-2
- record retrieval 21-35
- RECTVAL attribute 21-27
- RECTYPE attribute 21-27
- security 21-2
- segment attributes 21-15
- segment declarations 21-24
- SEGNAME attribute 21-15, 21-25
- SEGTYPE attribute 21-16
- server relationships 21-6
- SET READWOL command 21-35
- settings 21-36
- thread management 21-36
- unrelated files 21-12
- USAGE attribute 21-18

MODEL 204 DBMS 21-35

MODIFY operations for RMS 32-57

modifying synonyms for Enterprise Java Beans (EJB) 8-8

MQSeries Data Adapter 22-1 to 22-2

- Access Files 22-3
- application requests 22-1
- configuring 22-2
- CREATE SYNONYM command 22-4
- creating synonyms 22-3 to 22-4
- creating synonyms manually 22-3
- data formats 22-5
- environment variables 22-2
- FILEDEF command 22-2
- foreign messages 22-4
- Master Files 22-3
- metadata 22-2
- preparing the environment 22-2
- retrieving messages 22-4
- server family messages 22-4

Multifetch option for Adabas 2-74

multi-field JOINS for Adabas 2-85

multi-file Access Files for DATACOM 6-16

multi-file Master Files for DATACOM 6-15

multi-file structures for DATACOM 6-14, 6-18

multi-member sets for IDMS/DB 10-9

multiple connections for PeopleSoft 28-16 to 28-17, 28-22 to 28-24

multiple record types for RMS 32-17

multiply occurring fields for MODEL 204 21-5

Multi-Session Facility for Supra 36-4

multi-value (MU) fields for Adabas 2-62, 2-65, 2-67 to 2-68, 2-105

MUMPS Data Adapter 23-1 to 23-2

- Access Files 23-3
- application requests 23-1
- configuring 23-2
- connection attributes 23-2
- CREATE SYNONYM command 23-4
- creating synonyms 23-3 to 23-4
- creating synonyms manually 23-3

MUMPS Data Adapter (*continued*)

- data formats 23-6
- data values 23-5
- Master Files 23-3
- metadata 23-3
- preparing the environment 23-2
- redefining data 23-5
- security 23-2
- user authentication 23-2

My SQL Data Adapter

- data types 24-6

MySQL Data Adapter 24-1

- application requests 24-1
- connection attributes 24-2

N

National Language Support (NLS) for Microsoft SQL Server 19-8

Native RDBMS catalog 1-17

navigating files for Adabas 2-78

nested segment sets for RMS 32-26

network relationships for IDMS/DB 10-16

network subschema for IDMS/DB 10-48 to 10-49

NLS (National Language Support) for Microsoft SQL Server 19-8

node variables for SAP BW 33-21

Non-Aggregated Fields for Essbase 9-18

NONBLOCK mode 20-8, 29-17

- for Microsoft SQL Server Analytical Engine 20-8
- for Progress 29-17
- for Sybase ASE 37-19
- for Sybase IQ 38-19

non-consolidating members
handling in Essbase 9-19

non-descriptor fields for Adabas 2-84

non-relational data adapters 1-4

non-unique descendant segments for IDMS/DB
10-69, 10-83

Nucleus Data Adapter 25-1

- application requests 25-1
- AUTODISCONNECT command 25-6
- connection attributes 25-3 to 25-4
- connection scope 25-3, 25-5 to 25-6
- creating synonyms 25-6, 25-9
- creating synonyms manually 25-10
- creating synonyms using the Web Console 25-6
- data types 25-6
- default connections 25-3
- keywords 25-6
- metadata 25-6
- precision values 25-12
- preparing the environment 25-2
- preparing the environment on UNIX 25-2
- preparing the environment on Windows NT/
2000 and 25-2
- scale values 25-12
- SET AUTODISCONNECT command 25-5
- SET commands 25-14
- SET CONNECTION_ATTRIBUTES command 25-4
- SET DEFAULT_CONNECTION command 25-3
- set parameters 25-4, 25-14
- TIMEOUT command 25-14

null values 1-11

null-suppressed files for Adabas 2-17, 2-59, 2-86

numeric columns 11-13

- for IDMS/SQL 11-13
- for Informix 13-8
- for Microsoft Access 18-5
- for Microsoft SQL Server 19-8, 19-20
- for ODBC 26-5
- for Oracle 27-10
- for Progress 29-7
- for Sybase ASE 37-7

numeric values for XML 41-10

numeric variables for SAP BW 33-21

O

- OCCURS attribute 2-65
 - for Adabas 2-65, 2-69
 - for IDMS/DB 10-19
 - for MODEL 204 21-19
 - for RMS 32-24 to 32-25, 32-30
- OCCURS segment 10-26
 - for IDMS/DB 10-26
 - for MODEL 204 21-28
- OCCURS statement for RMS 32-56
- ODBC Data Adapter 26-1
 - Access File keywords 26-5
 - application requests 26-1
 - array retrieval 26-13
 - configuring 26-2
 - connection attributes 26-2
 - connection scope 26-2
 - CREATE SYNONYM command 26-5
 - creating synonyms 26-5
 - customizing 26-13
 - data types 26-5
 - database tables 26-5
 - mapping data types 26-5
 - metadata 26-5
 - numeric columns 26-5
 - preparing the environment 26-2
 - remote tables 26-5
 - SET commands 26-13
 - SET CONNECTION_ATTRIBUTES command 26-2
 - SET CONVERSION command 26-5
 - TIMEOUT command 26-13
- ODBC Driver Manager for Microsoft Access 18-2
- OLE/DB cubes for SAP BW 33-18
- OPEN attribute for Adabas 2-47
- open services for Bull 4-2
- Open/SQL support for SAP/R3 34-32
- operation considerations for Bull 4-2
- optimizing for Adabas on group fields 2-87
- Oracle Data Adapter 27-1
 - Access File keywords 27-10
 - application requests 27-1
 - array retrieval 27-25
 - authentication 27-5
 - block size 27-25
 - configuring 27-5
 - connection attributes 27-5
 - connection scope 27-5
 - connectivity with PeopleSoft 28-8
 - CREATE SYNONYM command 27-10
 - creating synonyms 27-10
 - creating synonyms using the Web Console 27-10
 - customizing 27-25
 - data types 27-10
 - Database Server 27-5
 - explicit authentication 27-5
 - index space 27-25
 - INSERTSIZE command 27-25
 - large character data types 27-10
 - metadata 27-10
 - numeric columns 27-10
 - password passthru authentication 27-5
 - precision values 27-21
 - preparing the environment 27-2
 - remote servers 27-2
 - scale values 27-21
 - security 27-5 to 27-6
 - SET AUTODISCONNECT command 27-5
 - SET CONNECTION_ATTRIBUTES command 27-5 to 27-6
 - SET DBSPACE command 27-25
 - specifying block size 27-25
 - stored procedures 27-25
 - synonyms 27-10
 - trusted authentication 27-5
 - using SQL Passthru 27-25
 - XA support 27-2
- Oracle data types 27-10

Oracle Database Server 27-5

ORDER field 2-69

for Adabas 2-69, 2-72

for IDMS/DB 10-29 to 10-30

for MODEL 204 21-20

for RMS 32-29

overriding parameters for IDMS/DB 10-91

Owner name 1-21

P

parallel segment sets for RMS 32-26

parameters for SET command 1-14

DEFCENT 1-15

YRTHRESH 1-15

PARENT attribute 21-16

for MODEL 204 21-16

for RMS 32-4, 32-51

PARENT keyword for IDMS/DB 10-18

parent/child relationship for RMS 32-4

parent/child view for Essbase 9-7 to 9-8, 9-12

partial fields in superdescriptors for Adabas 2-58

PASS attribute 2-53

for Adabas 2-53

for MODEL 204 21-25

PASSRECS for IDMS/SQL 11-18

password passthru authentication 19-3

for Microsoft SQL Server 19-3

for Oracle 27-5

for SAP BW 33-6

PDM data for Supra 36-15

PeopleSoft Data Adapter 28-1 to 28-2

adapter paths 28-12

additional functionality 28-42

administration reports 28-41

authentication 28-9

PeopleSoft Data Adapter (*continued*)

configuring 28-4, 28-15, 28-42

connection parameters 28-20

connectivity with Oracle connectivity 28-8

data location 28-3

DB2 Universal Database CLI adapter 28-8

debugging integration problems 28-42

environment settings 28-6

execution results 28-47

global settings 28-14

metadata 28-25 to 28-26, 28-35

modifying server start-up script in UNIX 28-7

multiple connections 28-16 to 28-17, 28-22 to 28-24

password required authentication 28-21

PeopleSoft PeopleTools 28-2

PeopleSoft RDBMS connectivity 28-6

PeopleTools Enterprise 28-2

preparing the environment 28-2

preparing the server environment 28-5

security 28-2, 28-13, 28-20, 28-36, 28-39 to 28-40, 28-45 to 28-46

security access 28-36

Security Administrator 28-25

server configuration 28-6, 28-12

server startup 28-11

stand-alone server usage 28-42

synonyms 28-21, 28-34 to 28-35

troubleshooting 28-48

TRUSTED authentication 28-21

UNIX environments 28-7

using with RDBMS 28-15

Windows environments 28-7

PeopleSoft PeopleTools 28-2

PeopleSoft RDBMS connectivity 28-6

PeopleTools Enterprise for PeopleSoft 28-2

periodic (PE) groups for Adabas 2-65, 2-105

periodic element (PE) groups for Adabas 2-62, 2-67 to 2-68

platform specific functionality for Adabas 2-71

Index

- POSITION attribute 10-19
 - for IDMS/DB 10-19
 - for RMS 32-28
- POSITION keyword for IDMS/DB 10-28
- precision values 13-17
 - for IDMS/SQL 11-13 to 11-14
 - for Informix 13-17
 - for Interplex 15-10
 - for Microsoft Access 18-14
 - for Microsoft SQL Server 19-20 to 19-21
 - for Nucleus 25-12
 - for ODBC 26-11
 - for Oracle 27-21
 - for Progress 29-15
 - for Sybase IQ 38-16
 - for Teradata 39-15
- Predict Dictionary for Adabas 2-14
- predict files in synonyms for Adabas 2-23
- Prefetch option for Adabas 2-74
- preparing the adapter environment 3-2
 - DATACOM 6-2
 - for Adabas 2-8
 - for Adabas on OpenVMS 2-8
 - for Allbase 3-2
 - for Bull 4-2
 - for C-ISAM 5-2
 - for Essbase 9-2
 - for IDMS/DB 10-2
 - for IDMS/SQL 11-2
 - for IMS 12-2
 - for Informix 13-20
 - for Informix on UNIX 13-2
 - for Informix on Windows NT/2000 13-2
 - for Interplex 15-13
 - for Lawson 17-2
 - for Microsoft Access 18-2
 - for Microsoft SQL Server Analytical Engine 20-2
 - for Microsoft SQL Server Data Adapter 19-2
 - for MODEL 204 21-2
 - for MQSeries 22-2
 - preparing the adapter environment (*continued*)
 - for MUMPS 23-2
 - for Nucleus 25-2
 - for ODBC 26-2
 - for Oracle 27-2
 - for Oracle on OS/390 and z/OS 27-2
 - for Oracle on UNIX 27-2
 - for Oracle on Windows NT/2000 27-2
 - for Oracle on z/OS 27-2
 - for Progress 29-2
 - for Red Brick 31-2
 - for Red Brick on UNIX 31-2
 - for Red Brick on Windows NT/2000 31-2
 - for Supra 36-2
 - for Sybase IQ 38-2
 - for Teradata 39-2
 - for Teradata on OS/390 39-2
 - for Teradata on UNIX 39-2
 - for Teradata on z/OS 39-2
 - for XML 41-2
- preparing the OpenVMS environment for Adabas 2-8
- preparing the UNIX environment for Adabas 2-8
- preparing the Windows NT/2000 environment for Adabas 2-8
- primary key 11-11
 - for IDMS/SQL 11-11
 - for RMS 32-12
- PRINT command for Essbase 9-12
- processing requests 1-7
- Progress Data Adapter 29-1
 - Access File keywords 29-7
 - application requests 29-1
 - array retrieval 29-17
 - configuring 29-3
 - connection attributes 29-3
 - connection scope 29-3
 - creating synonyms 29-7
 - creating synonyms using the Web Console 29-7

Progress Data Adapter (*continued*)

- customizing 29-17
- data types 29-7
- Database Server 29-3
- large character data types 29-7
- mapping data types 29-7
- metadata 29-7
- NONBLOCK mode 29-17
- numeric columns 29-7
- preparing the environment 29-2
- SET AUTODISCONNECT command 29-3
- SET CONNECTION_ATTRIBUTES command 29-3
- SET CONVERSION command 29-7
- TIMEOUT command 29-17

Progress data types 29-7

Progress Database Server 29-3

Q

QSYS and IFS file systems for J.D. Edwards 16-4

queries for SAP BW 33-8 to 33-10, 33-12

R

R/3 systems 34-11

Rdb Data Adapter 30-1

- Access File keywords 30-5
- application requests 30-1
- configuring 30-2
- connection attributes 30-2
- connection scope 30-2
- CREATE SYNONYM command 30-5
- creating synonyms 30-5
- creating synonyms manually 30-9
- creating synonyms using the Web Console 30-5
- data types 30-5
- default connection 30-4 to 30-5
- keywords 30-5
- metadata 30-5
- multi-version installation 30-2
- preparing the environment 30-2
- SET SERVER command 30-2

Rdb Data Adapter (*continued*)

- standard installation 30-2
- synonyms 30-8

RDBMS Data Adapter with PeopleSoft 28-15

Read Descriptor Value (L9) direct calls for Adabas 2-104

Read Logical calls for Adabas 2-98
2-106

Read Logical Navigation for Adabas 2-97

READ operations for RMS 32-35

Read Physical calls for Adabas 2-96, 2-98

READLIMIT keyword for Adabas 2-83

read-only fields for Microsoft SQL Server 19-8

record locking for RMS 32-34 to 32-35

- record retrieval 2-78, 10-87
 - for Adabas 2-78, 2-97
 - for IDMS/DB 10-67, 10-83 to 10-85, 10-87
 - for MODEL 204 21-35
 - for RMS 32-37
 - for SAP/R3 34-41 to 34-42
 - intervals for IDMS/DB 10-83
 - SAP BW 33-31

- record retrieval commands for DATACOM 6-22
 - GSETL and GETIT 6-22
 - SELFR and SELNR 6-22

record types for RMS 32-20 to 32-21, 32-30

RECORDLIMIT keyword for Adabas 2-83

RECTVAL attribute for MODEL 204 21-27

RECTYPE attribute for MODEL 204 21-27

RECTYPE field for RMS 32-17

Red Brick Data Adapter 31-1

- application requests 31-1
- AUTODISCONNECT command 31-2
- configuring 31-2

Index

- Red Brick Data Adapter (*continued*)
 - connection attributes 31-2 to 31-3
 - connection scope 31-2, 31-6
 - CREATE SYNONYM command 31-6
 - creating synonyms 31-6
 - creating synonyms using the Web Console 31-6
 - customizing 31-15
 - data types 31-6
 - default connections 31-2
 - metadata 31-6
 - precision values 31-13
 - preparing the environment 31-2
 - scale values 31-13
 - SET AUTODISCONNECT command 31-6
 - SET CONNECTION_ATTRIBUTES command 31-2 to 31-3
 - SET parameters 31-2
 - synonyms 31-6
- related record types for RMS 32-20
- relational data adapters 1-4
- Relational Gateway 1-18
- RELEASE attribute for Adabas 2-47
- release declaration in Access Files for Adabas 2-46
- REMARKS attribute 1-22
- REMOTE EXEC facility for BULL RTE 4-17
- REMOTE PUT/GET facility for BULL RTE 4-17
- remote servers 13-3
 - for Informix 13-3
 - for Oracle 27-2
 - for Sybase ASE 37-2
 - for Sybase IQ 38-2
- REMOTECAT command 7-29
- repeat groups for IDMS/DB 10-26
- repeating fields in Adabas files 2-62
- report library for J.D. Edwards 16-23
- reporting concepts for SAP BW 33-22 to 33-24
- reporting considerations for Adabas 2-78
- reporting from joined structures for IDMS/DB 10-78
- requests 1-7
- retrieval subtree for IDMS/DB 10-62 to 10-63, 10-79
- retrieval subtrees for IDMS/DB 10-65
- retrieving messages for MQSeries 22-4
- RMS Data Adapter 32-1
 - ACCEPT attribute 32-43
 - Access File examples 32-37
 - Access Parameter 32-34
 - ACTUAL attribute 32-9, 32-15, 32-44
 - AIS (Automatic Index Selection) 32-37 to 32-38
 - ALIAS attribute 32-5, 32-46
 - application requests 32-1
 - Automatic Index Selection (AIS) 32-37 to 32-38
 - commit processing 32-56
 - complex RMS keyed files 32-17
 - contiguous keys 32-13
 - DEFINE attribute 32-47
 - describing files manually 32-2
 - DESCRIPTION attribute 32-48
 - discontiguous keys 32-14
 - embedded data 32-30
 - embedded repeating data 32-23
 - FDL (File Description Language) 32-16
 - FIELD attribute 32-49
 - field attributes 32-4, 32-41 to 32-42
 - field names 32-5
 - FIELDNAME attribute 32-5, 32-49
 - FIELDTYPE attribute 32-49
 - FILE attribute 32-50
 - file attributes 32-2, 32-39
 - File Description Language (FDL) 32-16
 - file locking 32-34
 - FILENAME attribute 32-2, 32-50
 - FILESUFFIX attribute 32-52
 - FIX files 32-17
 - GROUP attribute 32-12 to 32-14, 32-50
 - INDEX attribute 32-49

RMS Data Adapter (*continued*)

- indexed files 32-12, 32-37
- KEY attribute 32-50
- keyed (indexed) files 32-12
- keyed RMS file 32-33
- keys 32-12
- LIBRARY file 32-20 to 32-21
- locked records 32-35
- LOCKMODE setting 32-35
- MAINTAIN operations 32-57
- manually describing files 32-2
- MAPFIELD value 32-31
- MAPVALUE field 32-31
- Master Files 32-39
- MODIFY operations 32-57
- multiple record types 32-17
- nested segment sets 32-26
- OCCURS attribute 32-24 to 32-25, 32-30
- OCCURS statement 32-56
- ORDER field 32-29
- parallel segment sets 32-26
- PARENT attribute 32-4, 32-51
- parent/child relationship 32-4
- POSITION attribute 32-28
- primary key 32-12
- READ operations 32-35
- record locking 32-34 to 32-35
- record retrieval 32-37
- record type indicator 32-30
- record types 32-20 to 32-21
- RECTYPE field 32-17
- related record types 32-20
- SEGMENT attribute 32-3, 32-51
- segment attributes 32-40
- segment name 32-12
- segments 32-3
- SEGNAME attribute 32-3, 32-12, 32-51
- SEGTYPE attribute 32-4
- single-field secondary keys 32-16
- SQL commands 32-57
- SQL COMMIT statement 32-56
- SQL operations 32-57

RMS Data Adapter (*continued*)

- SQL ROLLBACK statement 32-56
- STATMODE option 32-36
- SUFFIX attribute 32-2, 32-12, 32-52
- SYNONYM attribute 32-46
- syntax 32-39
- TITLE attribute 32-53
- unrelated record types 32-21
- USAGE attribute 32-6, 32-15, 32-54
- usage limitations 32-56

RMS FDL (File Description Language) 32-16

RMS File Description Language (FDL) 32-16

S

sample file descriptions for IDMS/DB 10-40

SAP BEx queries for SAP BW 33-13

SAP BW Data Adapter 33-1

- application requests 33-1
- authentication 33-6
- BEx Analyzer 33-8 to 33-9
- BW InfoCubes to OLE/DB cubes 33-18
- columnar reports 33-24
- configuring 33-5
- connection attributes 33-5 to 33-6
- creating dynamic dimensions 33-27
- creating synonyms 33-14 to 33-15
- cube-slicing logic 33-24
- dimension levels 33-21
- dynamic dimensions 33-28
- field names 33-13
- formulas 33-12
- hierarchies 33-27
- hierarchy variables 33-21
- InfoCubes 33-8 to 33-9
- key figures 33-11
- mapping metadata 33-19
- Master File 33-20
- member variables 33-21
- metadata 33-13, 33-17
- node variables 33-21

SAP BW Data Adapter (*continued*)

- numeric variables 33-21
- preparing the environment 33-2
- queries 33-8 to 33-10, 33-12
- record retrieval 33-31
- reporting concepts 33-22 to 33-24
- retrieving values 33-32
- SAP BEx queries 33-13
- security 33-6
- SET CONNECTION_ATTRIBUTES command 33-5
- Slice reports 33-24 to 33-25
- subcubes 33-8
- synonyms 33-14, 33-17
- user authentication 33-4, 33-6
- variable types 33-21, 33-25

SAP Synonyms for SAP/R3 34-18 to 34-19

SAP/R3 Data Adapter 34-1

- Access Files 34-18
- accessing multiple systems 34-11
- advanced features 34-33
- application requests 34-1
- BAPI support 34-33
- configuring 34-12
- connection attributes 34-12 to 34-13
- creating IDOC synonyms using the Web Console 34-21
- creating synonyms 34-18
- creating synonyms manually 34-28
- creating synonyms using the Web Console 34-19
- data types 34-31
- DOC Synonyms 34-18
- FM Synonyms 34-18, 34-23
- hierarchies 33-33 to 33-34
- IDOC Synonyms 34-21
- joins support 34-34
- logon requirements 34-11
- Master Files 34-18
- metadata 34-18
- Open/SQL support 34-32
- record retrieval 34-41 to 34-42

SAP/R3 Data Adapter (*continued*)

- SAP Synonyms 34-18 to 34-19
- secured requests 34-33
- SET CONNECTION_ATTRIBUTES command 34-12
- shared metadata 34-11
- synonyms 34-18, 34-27
- table support 34-30
- tracing 34-41

scale values

- for IDMS/SQL 11-13 to 11-14
- for Informix 13-17
- for Interplex 15-10
- for Microsoft Access 18-14
- for Microsoft SQL Server 19-20 to 19-21
- for Nucleus 25-12
- for ODBC 26-11
- for Oracle 27-21
- for Progress 29-15
- for Sybase IQ 38-16
- for Teradata 39-15

scenario dimensions for Essbase 9-9

screening conditions for IDMS/DB 10-81 to 10-82

screening criteria for SET ALL = ON command for IDMS/DB 10-71

security 9-2, 20-2

- for Essbase 9-2
- for IMS 12-7
- for Lawson 17-2, 17-6
- for Microsoft SQL Server 19-3, 19-5
- for Microsoft SQL Server Analytical Engine 20-2
- for MODEL 204 21-2
- for MUMPS 23-2
- for Oracle 27-5 to 27-6
- for PeopleSoft 28-2, 28-13, 28-20, 28-36, 28-39 to 28-40, 28-45 to 28-46
- for SAP BW 33-6
- for SAP/R3 34-33
- for Siebel 35-2
- for Supra 36-6 to 36-7
- J.D. Edwards 16-22

- Security Administrator for PeopleSoft 28-25
- security exit for Supra 36-6 to 36-7
- SEGMENT attribute for RMS 32-3, 32-51
- segment attributes 1-12, 2-47
 - for Adabas 2-47
 - for IDMS/DB 10-18
 - for MODEL 204 21-15
 - for RMS 32-40
- segment declaration keywords for IDMS/DB 10-34
- segment declarations 1-8 to 1-9, 1-12, 10-34
 - for IDMS/DB 10-34, 10-37
 - for IDMS/SQL 11-10
 - for MODEL 204 21-24
- SEGMENT logical record for Supra 36-14
- segment name for RMS 32-12
- segment retrieval for Adabas 2-78
- SEGNAM attribute for Adabas 2-49
- SEGNAME attribute 1-9, 1-12, 10-18
 - for IDMS 10-18
 - for MODEL 204 21-15, 21-25
 - for RMS 32-3, 32-12, 32-51
- SEGTYPE attribute 1-9, 10-18
 - for IDMS/DB 10-18
 - for MODEL 204 21-16
 - for RMS 32-4
- SELECT clause for IDMS/DB 10-15
- selection criteria 2-82
 - for Adabas 2-82
 - for IDMS/DB 10-65
- selective ALL prefix for IDMS/DB 10-72
- SEQFIELD attribute for Adabas 2-53
- SEQFIELD parameter for IDMS/DB 10-86
- Sequential Processing Facility (SPF) indexes for IDMS/DB 10-60
- server access for Siebel 35-2
- server configuration for PeopleSoft 28-6, 28-12
- server environment preparation 28-5
 - for PeopleSoft 28-5
 - for Siebel 35-3
- server family messages for MQSeries 22-4
- server file structure for Adabas 2-18
- server relationships for MODEL 204 21-6
- server settings for Bull 4-2
- server startup for PeopleSoft 28-11
- server start-up script in UNIX for PeopleSoft 28-7
- server terminology for DATACOM 6-5
- session commands for IDMS/SQL 11-3
- SET ALL = OFF command for IDMS/DB 10-70
- SET ALL = ON command for IDMS/DB 10-71
- SET ALL=OFF option for Adabas 2-80
- SET ALL=ON option for Adabas 2-81
- SET ALL=PASS option for Adabas 2-81
- SET AUTODISCONNECT command 8-5, 25-5
 - for Nucleus 25-5
 - for Oracle 27-5
 - for Progress 29-3
 - for Red Brick 31-6
 - for UniVerse 40-2
- SET commands
 - REMOTECAT 7-29
- SET CONNECTION_ATTRIBUTES command 4-7, 13-4
 - for Bull 4-7
 - for Enterprise Java Beans (EJB) 8-2
 - for Informix 13-4
 - for Microsoft Access 18-2
 - for Microsoft SQL Server 19-3
 - for Microsoft SQL Server Analytical Engine 20-4

Index

- SET CONNECTION_ATTRIBUTES command (*cont'd*)
 - for Nucleus 25-4
 - for ODBC 26-2
 - for Oracle 27-5 to 27-6
 - for Progress 29-3
 - for Red Brick 31-2 to 31-3
 - for SAP BW 33-5
 - for SAP/R3 34-12
 - for Siebel 35-2, 35-6
 - for Sybase ASE 37-3
 - for Teradata 39-3
 - for UniVerse 40-2
- SET CONVERSION command 13-8
 - for Essbase 9-12
 - for Informix 13-8
 - for ODBC 26-5
 - for Progress 29-7
 - for Sybase ASE 37-7
- SET DBSPACE command for Oracle 27-25
- SET DEFAULT_CONNECTION command 13-4
 - for Informix 13-4
 - for Nucleus 25-3
- SET FTPSERVER command for Bull 4-14 to 4-15
- SET MEASURE command for Essbase 9-10
- SET OPENMODE command for UniVerse 40-13
- SET parameters 1-14
 - YRTHRESH 1-15
- SET PASSWORD command for Adabas 2-10 to 2-11
- SET READWOL command for MODEL 204 21-35
- SET SCENARIO command for Essbase 9-9
- SET SERVER command for Rdb 30-2
- SET TRANSACTION command for IDMS/SQL 11-15
- SET TRANSACTIONS command for Microsoft SQL Server 19-23
- SET USER command for Bull 4-7
- set-based relationships for IDMS/DB 10-5
- set-based retrieval for IDMS/DB 10-88
- setting PSBs dynamically for IMS 12-9
- setup
 - J.D. Edwards 16-2
- Shared Members for Essbase 9-20
- short paths
 - for IDMS/DB 10-69
- short paths for IDMS/DB 10-68 to 10-69, 10-82 to 10-83
- Siebel Client 35-2
- Siebel Data Adapter 35-1 to 35-2
 - Access Files 35-10
 - application requests 35-1
 - Business Components 35-10
 - Business Objects 35-10
 - configuring 35-6
 - connection attributes 35-6 to 35-7
 - connection attributes from Web Console 35-8
 - creating synonyms 35-10
 - creating synonyms manually 35-14
 - data formats 35-17
 - Database Server 35-2
 - default connections 35-9
 - environment variables 35-3
 - Java 2 SDK 35-2 to 35-3
 - Java 2 SDK on UNIX 35-4
 - Java 2 SDK on Windows 35-3
 - listener 35-4
 - Master Files 35-10
 - metadata 35-9
 - preparing the adapter environment 35-2
 - preparing the server environment 35-3
 - security 35-2
 - server access 35-2
- SET CONNECTION_ATTRIBUTES command 35-2, 35-6
- Siebel Client 35-2
- Siebel Enterprise Server 35-8

- Siebel Data Adapter (*continued*)
 - Siebel Gateway Server 35-8
 - Siebel Object Manager 35-8
 - Siebel Object Server 35-7
 - Siebel Server 35-7
 - software requirements 35-2
 - synonyms 35-13
 - visibility mode 35-16
- Siebel Enterprise Server 35-8
- Siebel Gateway Server 35-8
- Siebel Object Manager 35-8
- Siebel Object Server 35-7
- Siebel Server 35-7
- simple FIND calls for Adabas 2-102, 2-106
- simple sets for IDMS/DB 10-6, 10-8
- single-field secondary keys for RMS 32-16
- Slice reports for SAP BW 33-24 to 33-25
- SLICEROPTIMIZATION command for Microsoft SQL Server Analytical Engine 20-9
- software requirements
 - J.D. Edwards 16-2
- software requirements for Siebel 35-2
- sort paths for IDMS/DB 10-74
- specifying file locations for C-ISAM 5-2
- SPF (Sequential Processing Facility) indexes for IDMS/DB 10-60
- SQL commands for RMS 32-57
- SQL COMMIT statement for RMS 32-56
- SQL operations for RMS 32-57
- SQL Passthru 13-21
 - for Informix 13-21
 - for Microsoft SQL Server 19-23
 - for Sybase ASE 37-22
- SQL requests 1-3
- SQL ROLLBACK statement for RMS 32-56
- SQL Translator 1-3
- stand-alone server usage for PeopleSoft 28-42
- standard field formats for Adabas 2-16
- STATMODE option for RMS 32-36
- stored procedures 13-21
 - for Informix 13-21
 - for Microsoft SQL Server 19-23
 - for Oracle 27-25
 - for Sybase ASE 37-22
- STRUCTURE record type for IDMS/DB 10-10
- subcubes for SAP BW 33-8
- subdescriptors (NOP) for Adabas 2-58
- subdescriptors for Adabas 2-19
- subschemas in IDMS/DB 10-36
- substitution variables for Essbase 9-24
- subtrees for Adabas 2-78
- SUFFIX attribute 1-8
- SUFFIX attribute for RMS 32-2, 32-12, 32-52
- SUM command for Essbase 9-12
- SUPEMPTY setting for Essbase 9-23
- superdescriptors for Adabas 2-19, 2-54 to 2-55, 2-72 to 2-73, 2-87
- SUPMISSING setting for Essbase 9-22
- Supra Central PDM 36-8
- Supra Data Adapter 36-1 to 36-2, 36-9
 - Access Files 36-13
 - adapter modules 36-15
 - application requests 36-1
 - Central PDM 36-4
 - compound non-unique data key 36-10

Supra Data Adapter (*continued*)

- compound unique data key 36-10
 - configuring 36-8
 - data structures 36-11
 - ddname CSIPARM 36-3
 - ddname CSISYSIN 36-3
 - EDASTART 36-3
 - embedded cross references 36-15
 - GATEKEPR Service Block 36-5
 - HEADER logical record 36-14
 - mapping considerations 36-9, 36-11
 - Master File 36-11
 - Master File cross references 36-15
 - metadata 36-11
 - module CFDP4001 36-15
 - module CFDP4002 36-16
 - module CFDP4003 36-16
 - Multi-Session Facility 36-4
 - PDM data 36-15
 - preparing the environment 36-2
 - security exit installation 36-6
 - security exit utilization 36-7
 - SEGMENT logical record 36-14
 - simple non-unique data key 36-10
 - simple unique data key 36-10
 - Supra Central PDM 36-8
 - Supra RDM 36-9
 - tracing facility 36-17
- Supra data structures 36-11
- Supra RDM (Relational Data Manager) 36-9
- Supra Relational Data Manager (RDM) 36-9
- SUPSHARE setting for Essbase 9-20
- SUPZERO setting for Essbase 9-22
- Sybase ASE Data Adapter 37-1 to 37-2
- Access File keywords 37-7
 - array retrieval 37-19
 - authentication 37-3
 - configuring 37-3
 - connection attributes 37-3
 - connection scope 37-3

Sybase ASE Data Adapter (*continued*)

- CREATE SYNONYM command 37-7
 - creating synonyms 37-7
 - creating synonyms manually 37-11
 - creating synonyms using the Web Console 37-7 to 37-8
 - customizing 37-19
 - data types 37-7
 - environment preparation 37-2
 - environment variables 37-2
 - large character data types 37-7
 - metadata 37-7
 - NONBLOCK mode 37-19
 - preparing the environment 37-2
 - remote servers 37-2
 - SET AUTODISCONNECT command 37-3
 - SET CONVERSION command 37-7
 - SQL Passthru 37-22
 - stored procedures 37-22
 - synonyms 37-7
- Sybase ASE data types 37-7
- large character 37-7
 - variable length 37-7
- Sybase interface A-4
- Sybase IQ Data Adapter 38-1
- Access File keywords 38-7
 - accessing remote servers 38-2
 - application requests 38-1
 - authentication 38-3
 - configuring 38-3
 - connection attributes 38-3
 - connection scope 38-3
 - CREATE SYNONYM command 38-7
 - customizing 38-19
 - data types 38-7
 - environment variables 38-2
 - metadata 38-7
 - NONBLOCK mode 38-19
 - preparing the environment 38-2
 - synonyms 38-7

SYNONYM attribute for RMS 32-46

synonyms 37-7

- for Adabas 2-22
- for Bull 4-10
- for DB2 7-11
- for Enterprise Java Beans (EJB) 8-6, 8-8
- for Essbase 9-4 to 9-5
- for IDMS/SQL 11-4
- for IMS 12-10, 12-15
- for Microsoft Access 18-5
- for Microsoft SQL Server 19-8, 19-11
- for Microsoft SQL Server Analytical Engine 20-6
- for MQSeries 22-3 to 22-4
- for MUMPS 23-3 to 23-4
- for Nucleus 25-6, 25-9
- for ODBC 26-5
- for Oracle 27-10
- for PeopleSoft 28-21, 28-34 to 28-35
- for Rdb 30-5 to 30-6, 30-8 to 30-9
- for Red Brick 31-6 to 31-7
- for SAP BW 33-14 to 33-15, 33-17
- for SAP/R3 34-18, 34-27 to 34-28
- for Siebel 35-10, 35-13
- for Sybase ASE 37-7 to 37-8, 37-11
- for Sybase IQ 38-7
- for Teradata 39-6 to 39-7, 39-9 to 39-10
- for UniVerse 40-2
- for XML 41-4

synonyms for superdescriptors in Adabas 2-73

syntax for RMS 32-39

T

table support for SAP/R3 34-30

TABlename attribute 1-12

tables 1-6

- documenting 1-22

Teradata Data Adapter 39-1

- application requests 39-1
- AUTODISCONNECT command 39-6

Teradata Data Adapter (*continued*)

- configuring 39-3
- connection attributes 39-4
- connection scope 39-6
- creating synonyms 39-6
- creating synonyms manually 39-10
- customization 39-17
- precision values 39-15
- preparing the environment 39-2
- scale values 39-15
- SET CONNECTION_ATTRIBUTES command 39-3
- set parameters 39-3, 39-6

thread management for MODEL 204 21-36

Time Series reporting for Essbase 9-17

TIMEOUT command 25-14

- for Bull 4-15
- for Nucleus 25-14
- for ODBC 26-13
- for Progress 29-17

TITLE ATTRIBUTE 1-25

TITLE attribute 1-21, 1-23 to 1-25, 32-53

TN FOCUS (Bull Data Adapter) 4-1

tracing facility 10-92

- for IDMS/DB 10-92
- for J.D. Edwards 16-23
- for SAP/R3 34-41
- for Supra 36-17

Transaction Coordinator for XA A-2 to A-3

- implementing A-3

troubleshooting for PeopleSoft 28-48

U

UDAs (User-Defined Attributes) for Essbase 9-8

uncommitted transactions (LUW) 26-2, 27-5, 37-3

- for Informix 13-4
- for Microsoft Access 18-2
- for Microsoft SQL Server 19-3

Index

unique descendant segments for IDMS/DB 10-69, 10-82

unique segments 2-79
for Adabas 2-79
for IDMS/DB 10-64 to 10-65, 10-68, 10-82

UniVerse Data Adapter 40-2
configuring 40-2
connection attributes 40-2
connection scope 40-2
CREATE SYNONYM command 40-2
creating synonyms 40-2
creating synonyms manually 40-2
customizing 40-13
Data Definition Language commands 40-13
data types 40-2, 40-11
declaring connection attributes using the Web Console 40-3
default connections 40-2
metadata 40-2
preparing the environment 40-2
SET AUTODISCONNECT command 40-2
SET CONNECTION_ATTRIBUTES command 40-2
SET OPENMODE command 40-13
synonyms 40-2

UNIX environments for PeopleSoft 28-7

unloading the CD-ROM
J.D. Edwards 16-3

unrelated files for MODEL 204 21-12

unrelated record types for RMS 32-21

USAGE attribute 1-9, 2-43
for Adabas 2-43
for IDMS/DB 10-22
for MODEL 204 21-18
for RMS 32-6, 32-15, 32-54
for XML 41-10

usage limitations for RMS 32-56

user authentication 35-2
for Essbase 9-2
for Microsoft SQL Server Analytical Engine 20-2
for MUMPS 23-2
for SAP BW 33-4, 33-6
for Siebel 35-2

User Requirements Table (URT) for DATACOM 6-2

User-Defined Attributes (UDAs) for Essbase 9-8

User-defined Metadata 1-18 to 1-19

V

value retrieval in SAP BW 33-32

variable length data types 13-8
for DB2 7-11
for Informix 13-8
for Microsoft SQL Server 19-8
for Oracle 27-10
for Progress 29-7
for Sybase ASE 37-7

variable types for SAP BW 33-21, 33-25

variable-length records in Adabas files 2-21, 2-62

virtual fields
creating 1-13 to 1-14

visibility mode for Siebel 35-16

VSAM files for C-ISAM 5-5

W

Web Application Server for Enterprise Java Beans (EJB) 8-2 to 8-3, 8-5

Web Console 40-3
configuring Adabas 2-9
configuring Adabas using OS/390 and z/OS parameters 2-9
configuring DATACOM 6-2
configuring IMS 12-9
connection attributes for SAP BW 33-6
connection attributes for SAP/R3 34-13

Web Console (continued)

- connection attributes for UniVerse 40-3
- creating synonyms for Bull 4-10
- creating synonyms for DB2 7-11
- creating synonyms for Enterprise Java Beans (EJB) 8-6
- creating synonyms for IMS 12-10
- creating synonyms for Informix 13-8
- creating synonyms for Microsoft Access 18-2
- creating synonyms for Microsoft SQL Server 19-9
- creating synonyms for Nucleus 25-6 to 25-7
- creating synonyms for Oracle 27-10
- creating synonyms for Progress 29-7
- creating synonyms for Rdb 30-5
- creating synonyms for Red Brick 31-6
- creating synonyms for SAP/R3 34-19
- creating synonyms for Sybase ASE 37-7 to 37-8
- creating synonyms in Progress 29-7
- modifying synonyms for Enterprise Java Beans (EJB) 8-8

WHERE clauses

- for Lawson 17-6

Windows environments for PeopleSoft 28-7**X**

XA configuration file A-4**XA Support 7-3**

- for DB2 7-3
- for Informix 13-4
- for Microsoft SQL Server 19-2
- for Oracle 27-2

XA Transaction Management feature 19-2, A-2

- compliant interfaces A-3
- for DB2 7-3
- for Informix 13-4
- for Microsoft SQL Server 19-2
- for Oracle 27-2

XML Data Adapter 41-1 to 41-2

- Access Files 41-4
- application requests 41-1
- configuring 41-2, 41-12
- creating synonyms 41-4, 41-6 to 41-7
- creating synonyms manually 41-4
- data formats 41-9
- data sources 41-7
- date fields 41-10
- environment variables 41-2
- FILEDEF command 41-2 to 41-3
- Master Files 41-2, 41-4
- metadata 41-4
- numeric values 41-10
- preparing the environment 41-2
- setting ACTUAL and USAGE attributes 41-10
- XML XFOCUS 41-11

XML data sources 41-7**XML XFOCUS 41-11**

- archiving XML documents 41-12
- executing XML documents 41-11

Y

YRTHRESH command 1-15**YRTHRESH parameter 1-15****Z**

zeros values

- suppressing in Essbase 9-22 to 9-23

Reader Comments

In an ongoing effort to produce effective documentation, the Documentation Services staff at Information Builders welcomes any opinion you can offer regarding this manual.

Please use this form to relay suggestions for improving this publication or to alert us to corrections. Identify specific pages where applicable. You can contact us through the following methods:

Mail: Documentation Services - Customer Support
Information Builders, Inc.
Two Penn Plaza
New York, NY 10121-2898

Fax: (212) 967-0460

E-mail: books_info@ibi.com

Web form: <http://www.informationbuilders.com/bookstore/derf.html>

Name: _____

Company: _____

Address: _____

Telephone: _____ Date: _____

E-mail: _____

Comments:

Reader Comments

Information Builders, Two Penn Plaza, New York, NY 10121-2898

(212) 736-4433

iWay Data Adapter Administration for UNIX, Windows, OpenVMS,
OS/400, OS/390 and z/OS Version 5 Release 2.3

DN3501109.0903